VŠB – Technická univerzita Ostrava

Fakulta elektrotechniky a informatiky

Katedra informatiky

Ovládací aplikace robotické platformy pro systém Android

Android Application for Robot Control

2015 Bc. Martin Stoklas

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student: **Bc. Martin Stoklas**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Ovládací aplikace robotické platformy pro systém Android
Android Application for Robot Control

Zásady pro vypracování:

Mobilní zařízení se v poslední době začínají využívat i jako ovládací konzole různých robotických platforem. Kromě řídící a pohonné jednotky jsou roboti vybaveni celou řadu senzorů a kamer. Příkladem může být velmi populární kvadrokoptéra AR.Drone a její ovládání na mobilních zařízeních se systémem Android. Cílem této práce bude vyvinou ovládací aplikaci pro pozemní roboty vybavené LIDARovým senzorem. Aplikace bude schopna využít data z LIDARu k detekci a lokalizaci překážek v prostoru, případně k lokalizaci vlastní platformy nesoucí LIDARový scanner.

1. Vypracujte přehled dostupných robotických platforem a komunikačních rozhraní.
2. Návrhněte a implementujte vlastního rozhraní pro komunikaci s vybranou platformu (např. Neato Robotics XV-15).
3. Nastudujte technologii LIDAR a po konzultaci s vedoucím práce vyberte vhodný lokalizační a detekční algoritmus.
4. Implementujte ovládací a vizualizační aplikaci pro platformu Android.
5. Otestujte a zhodnoťte Vaše řešení.

Seznam doporučené odborné literatury:

[1] Meier, R., Professional Android 4 Application Development, Wrox; 3 edition, 2012, ISBN 1118102274
[2] Murphy, M. L., Průvodce programováním mobilních aplikací, CPress, 2011, ISBN 978802513194
[3] Milette, G., Professional Android Sensor Programming, Wrox; 1 edition, 2012, ISBN 1118183487

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Mgr. Ing. Michal Krumnikl, Ph.D.**

Datum zadání: 01.09.2013

Datum odevzdání: 07.05.2015

doc. Dr. Ing. Eduard Sojka	prof. RNDr. Václav Snášel, CSc.
vedoucí katedry	*děkan fakulty*

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 29. dubna 2015 .....

podpis

Rád bych touto cestou poděkoval vedoucímu mé diplomové práce Mgr. Ing. Michalu Krumniklovi, Ph.D. za jeho odborné vedení a pomoc při zpracování této diplomové práce.

Abstrakt

Tato diplomová práce se zabývá ovládáním robotické platformy prostřednictvím vytvořené Android aplikace a dále problematikou lokalizace a mapování pomocí LIDAR technologie. Zpracován je stručný přehled robotických platforem a používaných komunikačních rozhraní. K testování implementované aplikace a SLAM řešení byl vybrán robotických vysavač Neato XV-15. Zvolené komunikační rozhraní pro komunikaci s robotickým vysavačem Neato XV-15 je v textu blíže specifikováno.

Práce rovněž obsahuje základní informace o technologii LIDAR a vybraných SLAM algoritmech a řešeních. Lokalizace a mapování probíhá pomocí zpracování dat získaných z laserového zařízení umístěného na robotovi. Pro samotné řešení problému simultánní lokalizace a mapování bylo vybráno BreezySLAM řešení.

Klíčová slova: LIDAR, SLAM, robot, platforma, lokalizace, mapování, komunikace

Abstract

This diploma thesis deals with control of robotic platform by created Android application and furthermore with localization and mapping issue by using LIDAR technology. Brief overview of robotic platforms and used communication interfaces is created in the thesis. Robotic vacuum cleaner Neato XV-15 was chosen for testing the implemented application and SLAM solution. Selected communication interface for communication with a robotic vacuum cleaner Neato XV-15 is more closely specified in the text.

The work also contains basic information about LIDAR technology and selected SLAM algorithms and solutions. Localization and mapping is performed by processing the data obtained from a laser system mounted on the robot. BreezySLAM solution was selected for solving the problem of simultaneous localization and mapping.

Keywords: LIDAR, SLAM, robot, platform, localization, mapping, communication

Seznam použitých symbolů a zkratek

API	–	Application programming interface
CSMA/CD	–	Carrier Sense Multiple Access with Collision Detection
CSV	–	Comma-separated values
EKF	–	Extended Kalman filter
I2C	–	Inter-Integrated Circuit
IMU	–	Inertial measurement unit
IP	–	Internet Protocol
GPS	–	Global Positioning System
GSM	–	Global Systém for Mobile Communications
LED	–	Light-emitting diode
LIDAR	–	Light Detection And Ranging
NDK	–	Native Development Kit
PGM	–	Portable Graymap Format
PTAM	–	Parallel Tracking and Mapping
QR	–	Quick Response
RANSAC	–	Random sample consensus
RPS	–	Room Positioning System
SLAM	–	Simultaneous localization and mapping
TCP	–	Transmission Control Protocol
UART	–	Universal Asynchronous Receiver / Transmitter
USB	–	Universal Serial BUS

Obsah

Seznam obrázků

1 Úvod

Robotické platformy se s postupem času stávají stále více oblíbené i mezi běžnými uživateli. Snadná dostupnost součástek a hotových řešení s přijatelnou cenou pomohla k masivnějšímu rozšíření těchto robotických systémů. S narůstajícím počtem robotů, především pak vzdušných robotů, je však o to více zapotřebí spolehlivého, nejlépe autonomního řízení a komunikace mezi roboty samotnými. V druhé kapitole této práce si představíme několik druhů robotických platforem a to jak pro pozemní použití, tak i pro letecké a podvodní účely. V třetí kapitole si pak stručně rozebereme základní komunikační rozhraní používané v robotických platformách.

Samozřejmě roboti nejsou jediným příkladem, kde nám jde o autonomní řízení, dalším příkladem mohou být automobily, které dokážou sami vykonávat činnost řízení, bez jakéhokoliv lidského zásahu. Avšak v obou případech je zapotřebí pro fungování autonomního řízení mít k dispozici informace o svém okolí a to v reálném čase. K získávání těchto dat slouží různá zařízení, jako jsou senzory, čidla, kamery apod. S těmito získanými daty je pak nutné příslušně nakládat, tedy zpracovávat je za účelem detekování překážek, plánování trasy atd. Známým problémem je simultánní lokalizace a mapování, jinak známý pod zkratkou SLAM, kterým tato práce bude zabývat ve čtvrté kapitole. Data jsou nejčastěji získávána pomocí LIDAR technologie, kde nejvýznamnější roli hraje laserové zařízení pro detekci a měření vzdálenosti objektů v okolí.

V páté kapitole rozebereme základní informace o robotickém vysavači Neato XV-15. Tento robot byl vybrán pro demonstraci vytvořené Android aplikace a to díky jeho laserovému zařízení, které ho dělá dobrou volbou při výběru hotového a poměrně levného robotického řešení pro řešení SLAM problémů. Samotná Android aplikace sloužící jak k ovládání robota, tak i k vizualizaci získaných LIDAR dat, bude popsána v šesté kapitole.

Poslední sedmá kapitola se zaměří na testování vytvořené Android aplikace a zhodnocení vybraného BreezySLAM řešení pro zpracování SLAM operací.

2 Robotické platformy

V této práci budu zmiňovat pouze pohyblivé roboty, které jsem rozdělil do tří základních skupin a to na pozemní roboty, roboty pro použití ve vzduchu a podvodní roboty.

2.1 Pozemní roboti

Mezi nejznámější pozemní roboty patří roboti s koly, kteří si svou oblíbenost získali hlavně díky své jednoduchosti, ceně a dostupných informací, které lze využít pro vytvoření svého vlastního robota. Pokud má být robot využíván ve složitějším terénu, lze použít pásový podvozek, čímž se předejde možnému prokluzování kol a umožní snadnější pohyb okolím. V neposlední řadě existují roboti s nohama (po vzoru lidí, či zvířat). Tito roboti mají vyšší pravděpodobnost, že se dostanou na požadované místo. Pro jejich komplikovanost, nutnost koordinovat pohyb apod. se ovšem musí počítat s výrazně vyšší cenou a nutností použít sofistikovaný řídící software.

V případě autonomie je zapotřebí roboty osadit různými čidly a senzory tak, aby dokázali reagovat na své okolí. Mezi důležitá čidla použitá v robotice patří čidla pro detekování překážek či pro hledání polohy a orientace objektu. V současné době jsou součástí robotů kamery, které mohou sloužit jak pro snímání obrazu, tak pro ovládání robota pomocí analýzy obrazu. Dražší variantou je použití laseru pro skenování svého okolí, k čemuž se dostaneme později.

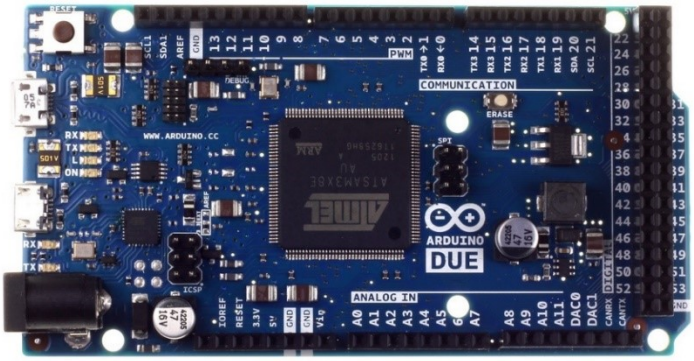
2.1.1 Arduino

Jedná se o úspěšnou open-source platformu založenou na mikrokontrolérech. Tato platforma nabízí levnou a jednoduchou cestu pro začátečníky, kteří by rádi začali zkoumat a využívat robotické možnosti. Arduino nabízí několik různých desek, které se dají použít pro vytvoření robota, dále nabízí volitelné komponenty, schémata zapojení, ukázky a hotová řešení.

Další výhodou této platformy je rozšířená komunita, jednoduché vývojové prostředí, softwarová i hardwarová rozšiřitelnost a v neposlední řadě svoboda pří výběru operačního systému.

Již v základu obsahuje Arduino knihovny pro komunikaci pomocí technologie Ethernet, GSM, Wi-Fi a nechybí ani I2C. Minimálně jeden sériový port pro komunikaci s dalším zařízením je vždy přítomen na Arduino desce. Ukázku Arduino Due desky můžeme vidět na obrázku 1. [1]

Základními senzory pro detekci překážek u této platformy jsou ultrazvukové dálkoměry, infračervené dálkoměry, LED s vysokým jasem a fotorezistory. Ovšem díky možnosti snadného rozšiřování není problém robota obohatit i o jiné senzory. [1]



Obrázek 1: Deska Arduino Due [1]

2.1.2 Roli

Roli je pozemní robot s pásovým typem podvozku. Robot je navrhnut tak, aby vydržel dlouhé trasy i obtížným terénem. Robot Roli je snadno přizpůsobitelný a to díky konstrukci, která je připravena pro připojení třinácti přídavných komponent a to konkrétně jedné zepředu, tří na zadní části konstrukce, čtyř na každém boku a jedné na horní části robota. Zadní část těla robota dále obsahuje odkládací prostor a odnímatelný držák na pití. Robota Roli můžeme vidět na obrázku 2. [2]

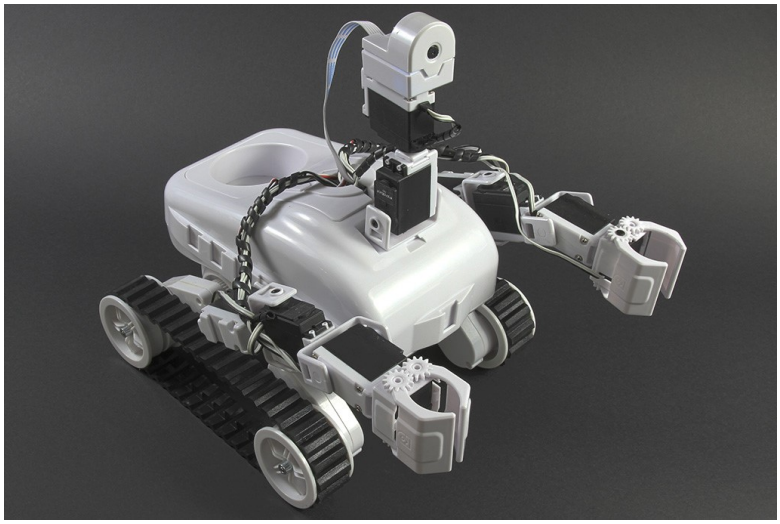
Řízení robota může být uskutečněno pomocí přítomného bezdrátového Wi-Fi připojení, dále jsou přítomny tři I2C porty, jeden port pro video kameru a tři vysokorychlostní UART porty. Přítomný je dále mikrofon, díky kterému je možné přijímat hlasové příkazy, a reproduktor pro reprodukci hlasu a hudby. [2]

Kamerový systém umožňuje pokročilé vizuální učení. Tedy robot je schopen naučit se nové pohyby jen např. pomocí sledování pohybu ruky. Dále nechybí funkcionalita pro detekování a sledování objektů, rozlišování objektů, obličejů, barev a v neposlední řadě čtení QR kódů. [2]

Robot je schopen uchopit menší předměty a posléze je přemístit či s nimi pracovat. Díky možnosti rozlišování objektů a jejich barev je tedy možné robotovi definovat, se kterým objektem chceme provést interakci a co přesně s ním má robot dělat. [2]

Ovládat robota lze snadno pomocí webového rozhraní. Díky integrovanému webovému serveru, který běží přímo v systému robota, se tedy stačí pouze připojit na jeho webové rozhraní. Poté má uživatel k dispozici ovládání pohybů robota, sledování obrazu z přítomné kamery atd. Dále pomocí existujícího EZ-Builder softwaru lze robota rozšiřovat o jeho schopnosti, jako jsou hlasové příkazy, pohyby, sledování objektů pomocí kamery apod. Software také umožňuje jednoduché vytvoření vlastních mobilních aplikací pro ovládání Roli robota. [2]

Není-li uživatel spokojen s dostupnými komponentami, má možnost využít EZ-Builder software k vytvoření své vlastní komponenty, která může být vytištěna na 3D tiskárně a sdílena s ostatními uživateli na komunitním fóru. [2]



Obrázek 2: Roli robot [2]

2.1.3 Atlas

Jedná se o humanoidního robota vytvořeného především pro průzkumné a záchranné účely. Příklad konstrukce robota můžeme vidět na obrázku 3. Výška tohoto robota se pohybuje kolem 1.8 metrů a váží přibližně 150 kilogramů. Konstrukce je vyrobená především z hliníku a titanu. Robot má k dispozici kamery a laserové zařízení, které se využívají např. k detekci objektů, navigaci terénem, lokalizací a mapováním objektů apod. K pohybu robota slouží jeho nohy (po vzoru lidí) a k pohybu obtížným terénem je schopen využít i nezávisle na sobě fungující ruce. Díky konstrukci podobající se lidem má tento robot vysoký potenciál využití, který je tak dnes limitován spíše umělou inteligencí a ovládacím softwarem. [3, 4]

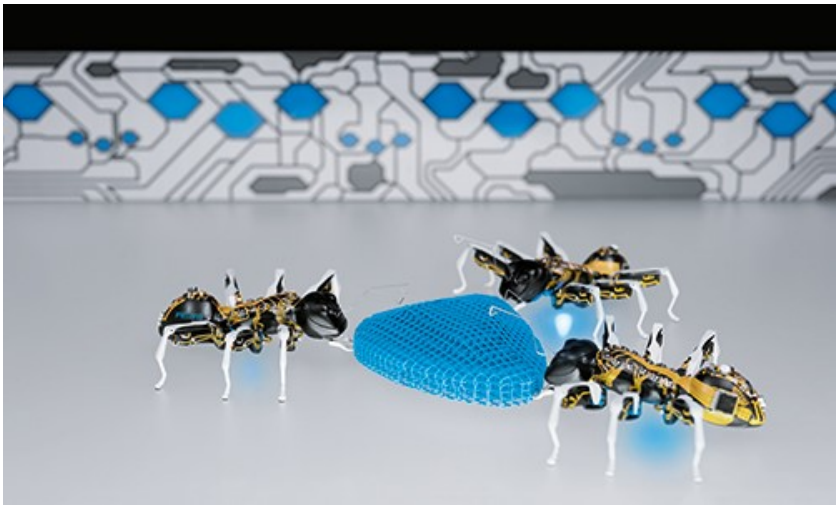
Provoz robota vzhledem k jeho robustné konstrukci a množství dostupných zařízení je velice náročný na zdroj elektrické energie. Proto jsou zde kladeny vysoké nároky na výdrž baterie, která slouží k napájení všech elektrických zařízení a pohybových systémů robota. S vyšší kapacitou baterie roste její váha a tedy nutnost pevnější konstrukce, proto se hodně práce na robotovi zaměřuje na redukci spotřeby elektrické energie. Z těchto důvodů je možné měnit nastavení robota, konkrétně tedy lze měnit tlak v pumpách potřebných pro provádění různých úkonů. Je tedy možné s robotem pracovat v úspornějším nastavení pokud víme, že robot nepotřebuje provádět žádné silově náročné operace. [4]

Obrázek 3: Humanoidní robot Atlas [3]

2.1.4 BionicANTs

Ukázkou malých robotů inspirovaných přírodou mohou být BionicANTs. Jedná se o malé roboty podobající se mravencům. Rozměry robota jsou mnohem větší, než jakých dosahuje skutečný mravenec. Pohybují se kolem 13.5 centimetrů na délku a 15 centimetrů na šířku. Hmotnost je přibližně 105 gramů. O pohyb robota se starají energeticky úsporné piezo-keramické motorky. K orientaci v okolí slouží stereo kamera a opticko-elektrický senzor. Robot je schopen komunikovat s ostatními umělými mravenci pomocí rádiové technologie. [5]

Hlavním účelem těchto robotů je spolupráce mezi sebou navzájem. Inspirováno skutečnými mravenci, díky většímu počtu jedinců je možné přemístit mnohonásobně těžší objekty, než jsou oni sami. Proto je zde snaha o uplatnění algoritmů, které při detekci objektu informují ostatní spolupracující roboty. Ti se poté společnými silami pokouší o přesun detekovaného objektu. Roboti jsou tedy neustále mezi sebou ve spojení a dohromady koordinují své akce a pohyby. Ukázku spolupráce robotů na společném úkolu lze vidět na obrázku 4. [5]



Obrázek 4: BionicANTs [5]

2.2 Vzdušní roboti

Široké možnosti využití představují vzdušní roboti. Ať již pro studijní nebo vědecké účely, armádu, průzkum terénu, rekreační využití, dnes velice diskutované doručování zboží, filmový průmysl, možnost střežení objektů atd.

Zřejmě nejvíce známým představitelem vzdušných robotů je dnes tzv. kvadroptéra. Ta se stala velice oblíbenou mezi řadou lidí a to nejen díky lehkému ovládání přizpůsobenému pro mobilní telefony, ale také díky své přijatelné ceně. Téměř všechny modely mají přimontovanou alespoň jednu kameru, díky které se může uživatel orientovat nebo jí použít pro oblíbené focení a nahrávání z jinak nepřístupných lokací.

Nevýhodou vzdušných robotů oproti pozemním je vyšší spotřeba energie, kvůli potřebě udržet se ve vzduchu. Proto je nutné zajistit bezpečné přistání v případě nedostatku energie pro pokračující let a také při ztrátě signálu. Pokud např. narazí do překážky odolnější pozemní robot, nemělo by to představovat žádný větší problém, pokud však vzdušný robot narazí do překážky a poškodí si při tom některý se svých mechanismů, bez kterých by nebyl schopen pokračovat v letu a zřítil by se z výšky dolů na zem, mohlo by dojít k jeho úplnému zničení.

Problém, který se v poslední době ukazuje, je nutnost řízení a komunikace vzdušných robotů mezi sebou. Je nepochybně žádoucí se touto problematikou zabývat, jelikož by v případě neřízeného provozu mohlo docházet k nebezpečným situacím. Dalším problémem je snadné porušení soukromí lidí, jelikož se vzdušný robot dokáže jednoduše dostat např. přes plot nad cizí pozemek a pořídit nežádoucí fotografie apod. Platný zákon č. 49/1997 Sb. o civilním letectví se vztahuje na modely o vzletové hmotnosti přesahující 20 kilogramů, tedy ve většině případů se tento zákon netýká právě malých a lehkých dronů. Ovšem byla již zpřesněna bezpečnost občanů například předpisem z 1. 3. 2012, zabývající se metodikou a postupy pro povolování leteckých prací prováděných bezpilotními systémy. Tedy existují již platná nařízení, která omezují použití vzdušných robotů. Vydání nových zákonů

či předpisů reflektujících současný stav a problémy spojené s širším využíváním bezpilotních letounů a vrtulníků běžnými lidmi, se očekává v příštích letech. [6, 7]

2.2.1 AR.Drone 2.0

Jedná se o velice populárního, rádiově řízeného, drona, kterého můžeme vidět na obrázku 5. Robot má pro komunikaci k dispozici USB port a Wi-Fi. Dále má k dispozici gyroskopy, akcelerometry, magnetometry, senzor pro měření tlaku, ultrazvukové senzory a videokamery. Pro měření rychlosti je použita kamera o frekvenci 60 snímků za sekundu. K autonomní navigaci se dají využít již hotová řešení, nejčastěji založená na PTAM vizuální navigaci. [8]

Do USB portu se dá připojit přídavný GPS modul, díky kterému lze na integrovanou paměť zaznamenávat letovou pozici a záběry z kamery. Dále přidává možnost plánování trasy letu. Tedy stačí robotovi zadat cílovou destinaci a robot se o let a návrat zpět již postará sám. [8]

Tohoto drona lze využít s moderními technologiemi, není tedy problém použití speciálních brýlí, které dokážou promítat obraz a sledovat přenos z kamery v reálném čase. K ovládání robota se dají použít vesměs všechny moderní zařízení od mobilních telefonů, tabletů, konzolí, počítačů až po nejmodernější náramky a brýle. [9]

Obrázek 5: AR.Drone 2.0 [8]

2.2.2 APM autopilot

Jinak také znám jako ArduPilot je univerzální platforma pro automatické řízení letadel, helikoptér (resp. kvadroptér), pozemních robotů a nejspíše brzy i vodních robotů. Jedná se o open-source řešení postavené na již zmiňované Arduino platformě. APM autopilot je rozdělen do tří kategorií a to:

- APM:Copter (helikoptéry),
- APM:Plane (křídlové letouny),
- APM:Rover (pozemní vozidla). [10]

Platforma vyniká v plně automatickém řízení, kdy např. u letadla se postará i o samotný vzlet a přistání. Dále disponuje sofistikovaným plánováním misí a ovládáním kamer, což se prokázalo v řadě soutěží, především však v pátracích a záchranných akcích. [11]

K dispozici jsou tři verze hardwaru pro automatické řízení a to Pixhawk, APM 2.5 / 2.6 a PX4 FMU. Záleží na uživateli, který hardware je pro něj ten nejvhodnější. [11]

Nabídka volitelného hardwaru, kterým může být robot osazen, je široká. Mezi nejdůležitějšími nechybí telemetrie, LIDARy a jiné senzory pro mapování okolí, kamery, GPS, kompasy, přídavné antény a v neposlední řadě senzory pro měření rychlosti. [11]

Ke komunikaci lze využít USB port, externí I2C port, dále se zde nachází port pro bezdrátovou telemetrii. Bohužel pro bezdrátovou komunikaci jako je Wi-Fi a nebo Bluetooth je třeba hledat externí řešení. [11]

2.2.3 eMotionButterflies

Další ukázkou přírodou inspirovaných robotů jsou umělí motýli eMotionButterflies. Jako v již dříve zmiňovaném případě robotických mravenců, ani tito motýli rozměrové neodpovídají těm skutečným. Rozpětí jejich křídel se pohybuje okolo 50 centimetrů, váha je však pouhých 32 gramů. Redukce váhy a minimalizace použitých součástek a materiálů bylo zapotřebí k docílení bezproblémového letu, protože pohyb tohoto robota je docílen pouze pohybem jeho dvou křídel. [12]

Robot disponuje inerciální měřící jednotkou, gyroskopem, akcelerometrem, kompasem a dvěma rádiovými moduly pro komunikaci s ostatními roboty v okolí a především pro komunikaci s centrálním počítačem. Centrální počítač se stará o zamezení kolizí mezi létajícími roboty. Kontroluje tedy všechny přítomné roboty a posléze patřičně upravuje a informuje konkrétní roboty o jejich dráhách letu. Detekce pozice každého robota je prováděna pomocí infračervených kamer, které snímají infračervené diody umístěné na tělech robotů. [12]

Na obrázku 6 lze vidět ukázku eMotionButterflies robotů.

Obrázek 6: eMotionButterflies [12]

2.3 Podvodní roboti

Podvodní roboti nemají tak širokou komunitu jako pozemní a vzdušní roboti. Informace pro vytvoření podvodního robota se proto shánějí obtížněji a také zde panují vyšší nároky na kvalitu konstrukce, součástek a celkového zpracování.

V současné době jsou tedy podvodní roboti používání spíše pro komerční, vědecké a studijní účely. Mezi širší veřejností nejsou tito roboti příliš využíváni.

Oproti vzdušným a pozemním robotům mají podvodní roboti nevýhodu v podobě obtížné bezdrátové komunikaci. Bezdrátová komunikace je ve vodě s přibývající hloubkou obtížná a proto se pro přenos dat nejčastěji používá kabelové spojení.

2.3.1 OpenROV

Jedná se o open-source platformu zejména pro studijní účely a prozkoumávání podvodního světa. Právě díky cílení na studijní sféru se cena pohybuje v přijatelných mezích.

K dostání je nejnovější verze robota OpenROV v2.7, kterou lze vidět na obrázku 7. Obraz z přední kamery se posílá do počítače přes dva tenké kabely. Robot disponuje LED osvětlením a dále lasery pro detekování překážek. [13]

Pro přidání dalšího hardwaru či techniky má robot na své konstrukci již předem vyhrazené místo. Dále obsahuje externí I2C port, RJ-45 konektor a mini USB. [13]

Obrázek 7: OpenROV 2.7 [13]

3 Komunikační rozhraní

Mezi nejpoužívanější komunikační rozhraní v robotech patří sériová komunikace. Nejčastěji je přítomná I2C sběrnice a USB port. Ve speciálních případech lze u robota nalézt klasický RJ-45 konektor pro Ethernet připojení, avšak moderní roboti v současné době kladou větší důraz na bezdrátovou komunikaci a to zejména pomocí Wi-Fi a nebo Bluetooth technologie.

Bezdrátové řešení má nespornou výhodu ve volnosti pohybu, není třeba se zatěžovat žádnými kabely apod. Avšak je třeba počítat s možností výpadku signálu, omezeným dosahem signálu a rušením.

Drátová komunikace je ve většině případů použita pouze za účelem testování, aktualizace firmwaru nebo zapojení dalšího zařízení, jako může být např. bezdrátové zařízení.

3.1 USB

USB je standard, který definuje komunikační protokoly, konektory a kabely použité pro spojení, komunikaci a napájení mezi zařízeními. Příkladem může být propojení počítače s periferiemi, jako je klávesnice, myš, tiskárna, digitální kamera, televizní karta, hard disk a jiná přenosná zařízení.

Dnes se USB využívá snad ve všech zařízeních, u kterých potřebujeme zajistit možnost komunikace s okolím a také může zároveň sloužit pro napájení a dobíjení baterie. Nejnovější specifikací je USB 3.1 dosahující přenosové rychlosti až 10 Gbit/s a s možností napájet zařízení s energetickou zátěží až do 10 W při dnes běžně používaném napětí 5 V (případně až 100 W při napětí 20 V). Očekává se rozšíření USB typu C, které se vyznačuje možností zapojení koncovky oběma stranami. [14]

3.2 I2C

Jedná se o sériovou sběrnici používanou pro připojování pomalejších periferií k procesoru počítače nebo vestavěného systému. V robotice se využívá sběrnice I2C především pro svou jednoduchost a levnou výrobu tam, kde není kladen důraz na rychlou přenosovou rychlost. V normálním režimu se přenosová rychlost pohybuje kolem 100 kbit/s, v nejrychlejším režimu dosahuje přenosová rychlost 3.4 Mbit/s. Přenosová rychlost v normálním režimu je však dostačující pro např. řízení stavu LED diod, zjišťování informací ze senzorů jako jsou akcelerometry, kompasy, gyroskopy, teploměry apod. I2C sběrnice si svou oblíbenost získala i díky jednoduchosti připojení více komponent na jednu sběrnici. [15]

3.3 Ethernet

Technologie Ethernet, z většiny standardizována jako IEEE 802.3, se v počítačových sítích používá pro komunikaci mezi dvěma a více zařízeními. Jako přenosové médium se používá kroucená dvoulinka nebo optické kabely. Nejpoužívanější přenosová rychlost v lokálních sítích se pohybuje mezi 100 Mbit/s až 10 Gbit/s. Vyšších přenosových rychlostí se dá dosáhnout již delší dobu, avšak nové standardy a síťové komponenty dosahující vyšších přenosových rychlostí se zatím příliš nerozšířily. [16]

Pro vytvoření větší sítě je zapotřebí použít aktivní prvky, které potřebujeme pro rozšíření dosahu, především ale pro směrování a propojení různých sítí mezi sebou. Mezi tyto aktivní prvky patří hub, switch, bridge a router.

O přístup zařízení k přenosovému médiu se stará CSMA/CD protokol, kde principem je vysílání dat stanicí a jejím současným naslouchánímm, zda nezachytí jiné vysílání, čímž by došlo ke kolizi. Pokud ke kolizi dojde, vysílání se o náhodnou dobu odloží a pak opakuje. [16]

3.4 Wi-Fi

Wi-Fi představuje technologii pro bezdrátovou komunikaci v počítačových sítích. Skupina standardů značených IEEE 802.11 popisují právě bezdrátovou komunikaci a definují použitá pásma, druh použité modulace rádiového signálu, protokoly, zabezpečení atd.

Tato technologie využívá ultra vysokou frekvenci 2.4 GHz a super vysokou frekvenci 5.4 GHz rádiového pásma. Za využívání těchto frekvenčních pásem není dnes třeba platit žádné licenční poplatky. [17]

Hlavním účelem vzniku této bezdrátové technologie bylo oproštění se od nutnosti používání kabelové infrastruktury pro lokální počítačové sítě. I když dnes se často setkáváme i s mnohem rozsáhlejšími Wi-Fi sítěmi, které nenahrazují pouze ty lokální.

3.5 Bluetooth

Jedná se o bezdrátovou technologii pro přenos dat na krátkou vzdálenost. Pro přenos dat se používají rádiové vlny o ultra vysoké frekvenci od 2.4 GHz po 2.485 GHz. Hlavním důvodem, proč tato technologie vznikla, byla snaha o vytvoření bezdrátové alternativy k drátové RS-232 komunikaci. [18]

Pomocí technologie Bluetooth je možné připojit více zařízení najednou, avšak svou popularitu tato technologie získala i s „pouhým“ propojováním dvou zařízení mezi sebou. Příkladem může být připojení bezdrátových sluchátek s mobilním telefonem, nebo propojení mobilního telefonu s palubním počítačem v automobilu.

Nejnovější specifikací je Bluetooth verze 4.2. Již od verze 4.0 se tato technologie chlubí nízkou energetickou náročností a poměrně vysokou přenosovou rychlostí a to kolem 24 Mbit/s. Dosah signálu dostačujícího ke komunikaci se v běžných podmínkách pohybuje kolem 10 metrů. Pro průmyslové účely může být dosah navýšen až na 100 metrů a to pomocí vyššího vysílacího výkonu. [18]

4 LIDAR technologie

Tato technologie se používá k detekci objektů a měření vzdálenosti mezi snímacím zařízením a detekovaným objektem. Pro tyto účely se využívá světelný paprsek, který je vysokou rychlostí emitován laserovým zařízením. Tento paprsek je po odrazu od povrchu objektu zachycen detektorem. Poté dojde k výpočtu vzdálenosti od detekovaného objektu, což se dá vyjádřit jednoduchým výpočtem, kdy vezmeme čas, který paprsek urazil k objektu a cestou zpět, tento čas vynásobíme rychlostí světla a vydělíme dvěma (cesta tam a zpět). Jedná se pouze o obecný popis, samotný výpočet se samozřejmě řídí dalšími okolnostmi. Např. pokud se měření provádí letecky, je zapotřebí brát v úvahu polohu letadla, o což se stará GPS, a dále pohyb a natočení letadla, k čemuž se používá IMU. Jedná se o zařízení pro měření orientace, rychlosti a gravitačních sil za pomoci gyroskopů, akcelerometrů a případně také magnetometrů. [19, 20, 21]

Technologie LIDAR má široké využití, nejčastějším příkladem jsou autonomní vozidla a roboti, vytváření map z nasnímaného terénu, budov, vegetace apod. Uplatnění se najde v mnoha oborech jako je archeologie, seismologie, geografie, geologie, atmosférické fyziky, meteorologie, lesnictví, automobilový průmysl, robotika a v neposlední řadě se tato technologie využívá pro armádní účely jako je např. navádění střel, detekování cílů, ovládání automatický zbraní atd. [21]

Mezi základní komponenty pro fungování LIDAR systému patří laser. Nejčastěji použité vlnové délky jsou 532 nm (jinak také známé jako zelené světlo), které dokáže proniknout vodou a 1064 nm (téměř infračervené světlo), nejčastěji využívané pro mapování terénu. Další využívanou vlnovou délkou je 1550 nm a to především ve vědecké a armádní sféře. Vyšší vlnové délky jsou využívané především na delší vzdálenosti, avšak s menší přesností. Dalšími použitými komponenty v tomto systému je optika ovlivňující detekovanou vzdálenost a natočení a dále detektor záření pro detekci elektromagnetického vlnění. V některých případech jako jsou satelity a letadla, je zapotřebí znát absolutní pozici a orientaci senzoru a proto jsou využívány již dříve zmíněné GPS zařízení a IMU. [20, 22]

LIDAR technologie umožňují mapování a lokalizaci objektů jak ve 2D, tak i 3D prostoru. Nejčastějším příkladem užití ve 2D je mapování místností, budov a okolí, kde si vystačíme pouze s informacemi, kde se objekty nachází. Ukázku mapování místnosti můžeme vidět na obrázku 8. Oproti tomu již náročnější 3D mapování lze využít takřka kdekoliv, díky preciznímu mapování a vizualizaci reálného světa.

Obrázek 8: Využití laserového zařízení k detekci objektů [23]

4.1 Lokalizace a mapování

Pro autonomní ovládání např. robotů a vozidel je zapotřebí vyřešit problém simultánní lokalizace a mapování (SLAM). Jedná se o tvoření nebo aktualizaci mapy předem neznámého prostředí a neustálé sledování pozice ovládané jednotky. Sledování pozice ovládané jednotky je zapotřebí pro pozdější plánování trasy, zaznamenávání trajektorie, správné generování mapy apod. Mezi nejznámější používané metody pro řešení této problematiky patří tzv. částicový filtr a rozšířený Kalmanův filtr.

4.1.1 Částicový filtr

Jedná se o výpočetně efektivní a robustní metodu, která se dá využít jak pro lokalizaci (např. robota), tak i mapování. Princip metody spočívá ve „vhodně“ umístěných vzorcích. Vzorky jsou umístěny náhodně a posléze jsou ohodnoceny váhou. Váha vzorku je ovlivněna pohybem, úhlem natočení robota a šumem. [24, 25]

Nevýhodou částicového filtru je neschopnost řešit tzv. „problém uneseného robota“. Tento problém se vyskytne v případě, kdy je robot, nejčastěji ručně, přemístěn. Tím nastane situace, kdy váhy vzorků nebudou již aktuální a nebude je možné ani znovu vypočítat, jelikož se kolem nové pozice robota nebudou žádné vzorky vyskytovat. Z tohoto důvodu se při každé iteraci mohou rovnoměrně přidávat náhodné vzorky s nízkou váhou, aby bylo možné se z takovéto situace dostat a znovu se v prostředí zorientovat. [24]

Dalším problémem je přepočítávání vah vzorků. Pouze vzorky ve shodě s mapou a získanými daty ze senzorů získají větší váhu. Tím nastane situace, kdy většina vzorků bude mít zanedbatelnou váhu, čemuž je třeba předejít. Proto se používá tzv. převzorkování, kdy vzorky s větší váhou budou do množiny zařazeny vícekrát a naopak se odstraní vzorky se zanedbatelnou váhou. Tímto dojde k vytvoření shluku vzorků, které určí aktuální pozici robota. [24, 25]

4.1.2 Orientační body

Orientační či jiné význačné body v prostředí se používají k lokalizaci robota. Takovými body mohou být objekty v prostoru, rohy zdí a případně se orientační body mohou tvořit i podél dlouhé zdi, např. každých 50 cm apod. Orientační body by měly být snadno rozpoznatelné z různých pozic robota a také odlišných úhlů naklonění. Body by neměly být příliš blízko u sebe, aby se předešlo snadnému zaměnění bodů mezi sebou. Snahou by tedy mělo být jednoznačné rozlišení orientačních bodů za jakýchkoliv situací. Důležité je také mít dostatek orientačních bodů ve skenovaném prostředí, aby robot vždy viděl alespoň některé význačné body, podle kterých se dokáže v prostoru orientovat. V žádném případě by se jako orientační body neměly používat pohyblivé objekty.

4.1.2.1 RANSAC

RANSAC je metoda, která může být využita k extrakci přímek z laserového měření (skenování). Tyto přímky mohou být posléze použity jako orientační body v prostoru. Postup fungování RANSAC metody spočívá v náhodném výběru vzorků z laserového měření a použití metody nejmenších čtverců k nalezení nejlepší pozice přímky procházející skrze detekované body. Poté již pouze dojde ke kontrole počtu bodů, které leží blízko vytvořené přímky. V případě, že počet bodů bude dostačovat (bude splňovat nastavené minimum), lze prohlásit, že detekovaný objekt můžeme reprezentovat vytvořenou přímkou. [26]

4.1.3 Kalmanův filtr

Kalmanův filtr představuje řešení pro lineární filtrování diskrétních dat. Princip fungování je založen na rekurzy, tedy algoritmus vychází ze svého předchozího stavu a nepotřebuje si uchovávat všechny předchozí data. Kalmanův filtr reprezentuje hustotu odhadu polohy objektu dvěma parametry a to střední hodnotou a rozptylem. Tyto parametry reprezentují vrchol a šířku Gaussovy křivky. [27, 28]

Metoda počítá se šumem na vstupu, který ovlivňuje výstupní odhad pozice. V algoritmu se tedy počítá s předchozí důvěryhodností výsledku, která je použita pro korekci nového odhadu. Pro použití filtru k odhadu pozice v nelineárním systému, jako je případ lokalizace robota, je nejprve zapotřebí algoritmus pozměnit. Nejznámějším upraveným filtrem se stal tzv. rozšířený Kalmanův filtr, jinak také znám pod zkratkou EKF. [27, 28]

4.1.4 GraphSLAM

Algoritmus lze použít jak pro řešení lokalizace tak i mapování. Základní technika je založená na měření vzdálenosti od orientačních bodů v okolí a pomocí těchto bodů získat odhad o aktuální pozici pozorovaného objektu. Informace získané z dalších skenování okolí dále pomáhají upřesnit pozice zadaných orientačních bodů a tím i zpřesnění odhadu pozice objektu. [29]

GraphSLAM algoritmus pro řešení simultánní lokalizace a mapování je považován za poměrně přesný a rychlý. Reprezentace dat za pomocí grafů umožňuje zpracování poměrně rozsáhlých dat a princip fungování je celkem jednoduchý k pochopení a implementaci. [29]

4.1.5 CoreSLAM

CoreSLAM, jinak také znám jako tinySLAM, je velice oblíbený pro svou jednoduchost a rychlost, poměrně snadné pochopení a další rozšiřitelnost či použití v jiném sofistikovanějším řešení. Vysoké rychlosti se snaží dosáhnout efektivním zpracováním dat a pro co nejrychlejší výpočty se snaží pracovat pouze s celými čísly. [30, 31]

Myšlenkou CoreSLAM řešení je integrování informací z laserového zařízení do lokalizačního subsystému, který je postaven na částicovém filtru. Pro řešení lokalizace se zde nachází dvě hlavní funkce a to *ts_distance_scan_to_map* a *ts_map_update*. Funkce *ts_distance_scan_to_map* se chová jako pravděpodobností funkce používaná k testování všech pozičních hypotéz (částic, resp. vzorků) ve filtru. K tomu dochází jednoduchým sečtením všech hodnot v mapě s odpovídajícími body v provedeném skenování (relativně od pozice částice, resp. vzorku). Tato metoda je zpracována velice rychle a to díky využívání pouze celočíselných operací, což dovoluje zpracovávat vysoký počet částic, resp. vzorků, v reálném čase. [30]

Funkce *ts_map_update* je použita k tvoření mapy v závislosti na pohybu robota. Generování mapy kompatibilní s částicovým filtrem není zcela přímočaré. Vrchol pravděpodobností funkce je velice důležitý pro efektivitu filtru a z tohoto důvodu by měl být jednoduše kontrolovatelný. Této skutečnosti je docíleno použitím mapy formátu PGM (využívají se různé stupně šedi). Aktualizace mapy probíhá vytvářením černých míst na mapě korespondujících vrcholům pravděpodobností funkce. Pro každou detekovanou překážku algoritmus nevykresluje pouze jeden bod, ale vyznačuje místo kolem překážky, takže nejintenzivnější odstín šedé barvy se nachází na místě detekovaného objektu. Mapa je aktualizována po poslední aktualizaci pozice částicového filtru (což je vážený průměru všech částic, resp. vzorků, po vyhodnocení pravděpodobností funkce). [30]

Další funkcí, použitou při aktualizaci mapy, je *ts_map_laser_ray*. V ní se uplatňuje Bresenhamův algoritmus, pro vykreslení laserových paprsků do mapy, s dalším vylepšeným Bresenhamovým algoritmem k výpočtu správné struktury. Opět se používají pouze celočíselné operace s výjimkou kritické části při dělení. [30]

Díky postupu zvýrazňování (ztmavování) detekovaných překážek v mapě a naopak zesvětlovaní míst, na kterých nebylo ve zpracovávaném skenování nic detekováno, lze na mapě patřičně reflektovat změny v okolí, jakými mohou být i nežádoucí pohyblivé objekty. Dále díky tomu, že algoritmy pracují pouze s jedinou mapou, bylo docíleno snížení paměťových nároků. Většina ostatních řešení fungujících na podobném principu jako je CoreSLAM používají alespoň dvě mapy. [30, 31]

Pro lokalizaci robota v mapě je použit jednoduchý Monte-Carlo algoritmus. Lokalizace probíhá výpočtem nové predikované pozice robota a porovnáním aktuálně naskenovaného okolí s vytvořenou mapou. Pokud je nalezen přesnější odhad pozice v aktuálně naměřených datech, dojde k aktualizaci pozice robota. [30, 31]

CoreSLAM je často porovnáván s FastSLAM a DP-SLAM řešením. Tyto jmenované SLAM řešení používají algoritmy vycházející z Kalmanových a částicových filtrů, avšak CoreSLAM vyniká svou jednoduchou a poměrně efektivní implementací. Ostatní řešení jsou mnohem komplexnější

a složitější na pochopení. Ke zpřesnění mapování a lokalizace je možné algoritmu poskytnout data z odometrie, nicméně CoreSLAM dokáže fungovat, v závislosti na prostředí a rychlosti pohybu robota, vcelku dobře i bez těchto údajů. [30, 32]

4.1.6 BreezySLAM

Jedná se o open-source SLAM řešení postavené na základě CoreSLAM. Tento projekt byl primárně vytvořen pro použití v jazyce Python, avšak je rozšířen i pro Matlab, C++ a jazyk Java. O řešení SLAM problému se stará samotný CoreSLAM. BreezySLAM tedy tvoří spíše mezivrstvu pro použití ve více programovacích jazycích a přináší možnost využití grafického rozhraní. Dále usnadňuje nastavení CoreSLAM parametrů, které jsou důležité pro správné fungování algoritmů a vrácení zpracovaných výsledků. [33, 34]

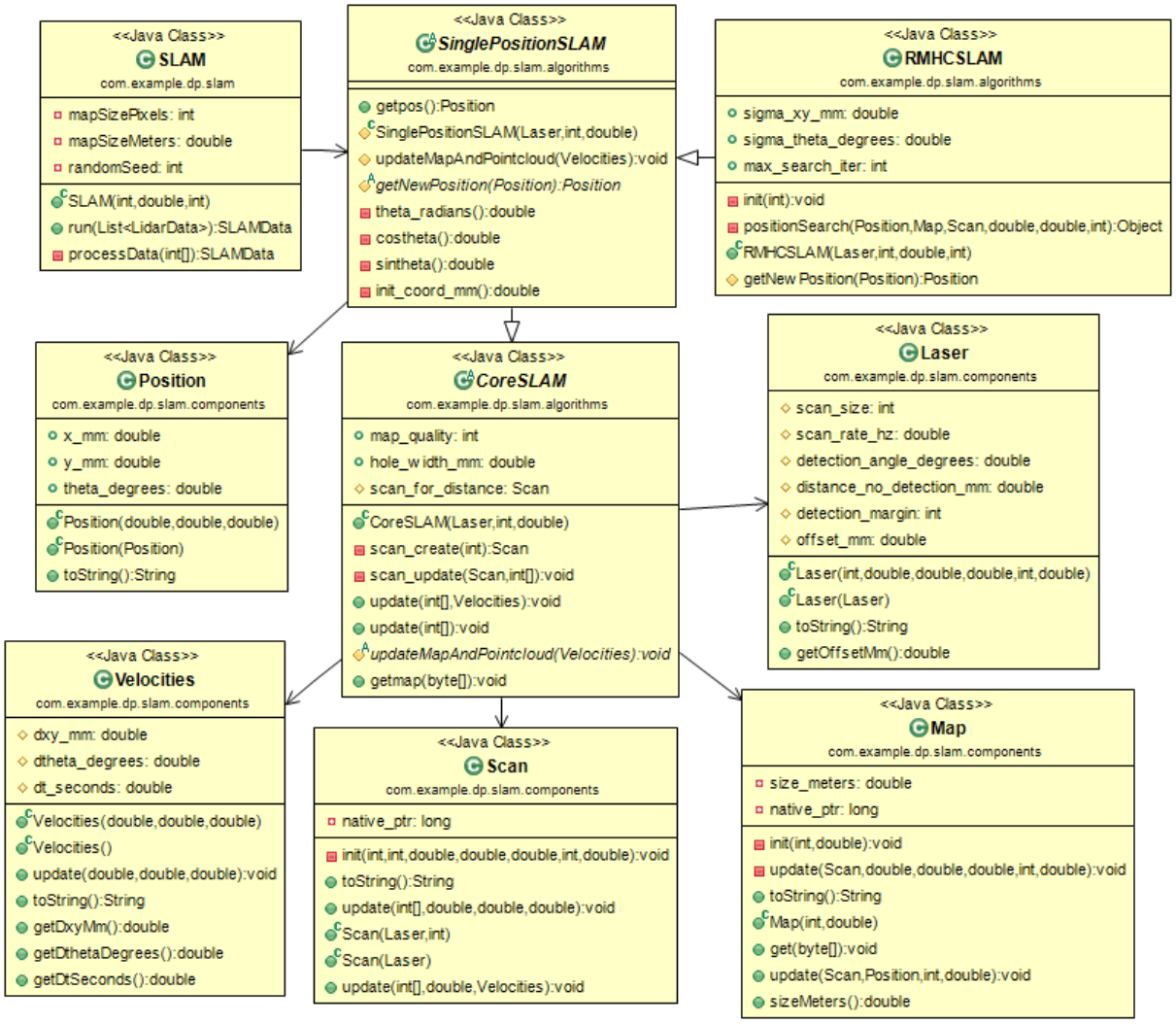
Třída *SLAM* je základní třídou starající se o zpracování příslušných algoritmů. Důležitými parametry předávajícími se v konstruktoru této třídy jsou velikost mapy v pixelech a měřítko mapy v metrech. Metoda *run* se stará o výběr relevantních dat zaslaných v parametru metody a vrací vytvořenou mapu společně s aktuálně vypočítanou pozicí robota. O samotné volání metod pro zpracování SLAM operací se stará metoda *processData*.

Pro korektní práci algoritmů je zapotřebí správně nastavit instanci třídy *Laser* a to pomocí parametrů zaslaných v konstruktoru třídy. Prvním parametrem je počet paprsků v jednom skenování (měření). Druhý parametr značí frekvenci prováděných skenování a třetí počet detekovaných úhlů. Čtvrtým parametrem se určuje počet paprsků na okrajích skenování, které mají být ignorovány, a poslední pátý parametr udává odsazení laseru od středu robota.

Třídy *DeterministicSLAM* a *RMHCSLAM* dědí z abstraktních tříd *CoreSLAM* a *SinglePositionSLAM*, které definují základní metody zpracovávající úkony lokalizace a mapování. Třída *DeterministicSLAM* se využije v případě, kdy máme k dispozici údaje z odometrie, která pomůže přesnějšímu odhadu pozice robota. V případě, že informace z odometrie nemáme k dispozici či jí prostě nechceme využít, použijeme třídu *RMHCSLAM*. Obě jmenované třídy v konstruktoru očekávají instanci třídy *Laser* a informace o velikosti a měřítku mapy, aby mohly korektně zpracovávat požadované úkony.

Další třídy jsou zde především pro ukládání informací, případně pro manipulaci s daty. Jako např. třída *Position* slouží pro práci s pozicí robota. Třída *Map* pro aktualizaci, získávání a uchovávání dat o mapě. Třída *Scan* se používá při práci s naskenovanými daty, tedy s informacemi o nalezených překážkách, jejich vzdálenosti atd.

Vazby a přehled základních tříd a metod lze vidět na obrázku 9.

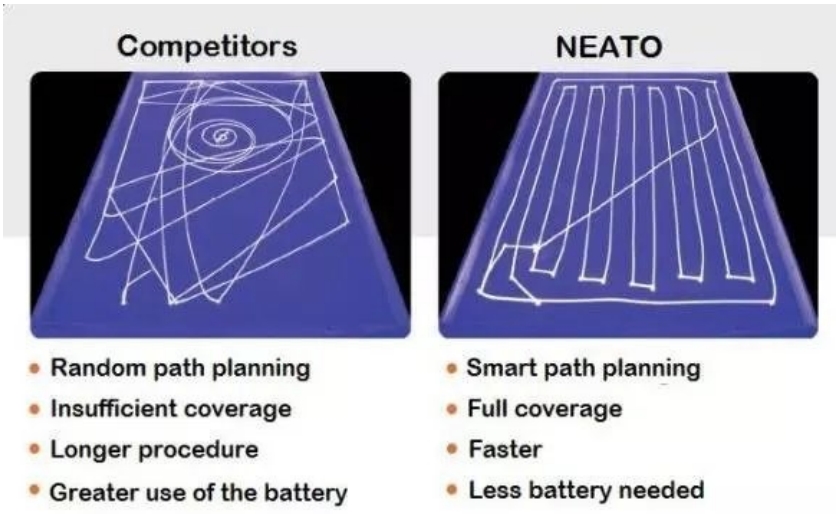


Obrázek 9: Třídní diagram základních BreezySLAM tříd

5 Neato XV-15

Tento robotický vysavač je evropskou verzí originálního modelu Neato XV-11. Oba modely si získaly širokou pozornost a oblíbenost díky disponujícímu laseru, který slouží k detekci překážek. Spousta robotických nadšenců si tedy tento robotický vysavač pořizovala jen se záměrem získat za vcelku přijatelnou cenu právě laserové zařízení, které by samo o sobě bylo dražší.

Díky využití laserového řešení k detekci překážek nabízí tento robot chytré plánování trasy úklidu, čímž se docílilo lepšího celkového pokrytí prostoru s rychlejším a energeticky méně náročným vysáváním. Technologie pro plánování cesty, která je u toho robota použita, se nazývá RPS (Room Positioning System) a ukázku jejího fungování lze pozorovat na obrázku 10. Oproti konkurenčním řešením, skládajících se většinou z náhodného plánování trasy, se jedná o velký krok kupředu. [23]



Obrázek 10: Ukázka chování RPS technologie [23]

Tento Neato vysavač však nemusí sloužit pouze pro domácí úklid. Robota je možné ovládat pomocí příkazů zasílaných přes přítomný USB port. Lze tedy s robotem pohybovat, zjišťovat jeho aktuální stav, ale to nejpodstatnější, lze získávat údaje z laserového snímače. Není se tedy čemu divit, že tento robotický vysavač spousta lidí nepoužívá pro svůj původně zamýšlený účel, ale spíše jako nástroj k dalšímu testování, bádání a práci s LIDAR technologií.

5.1 Rozhraní pro komunikaci s robotem

Veškerá komunikace včetně zasílaných příkazů musí být uskutečněna přes USB port umístěný na boku robota. Pro lepší a pohodlnější použití robota byl proto přidán router TP-Link TL-WR703N, který zajišťuje bezdrátovou komunikaci. Na routeru běží linuxová distribuce OpenWrt typicky určená pro tzv. vestavěné zařízení (embedded devices), kde nejčastějším příkladem jsou právě bezdrátové routery. O napájení tohoto routeru se stará baterie robota. Pro komunikaci mezi TCP síťovým připojením a sériovým USB portem je použit svobodný software s názvem ser2net. Samotná USB komunikace je řízená standardním linuxovým ovladačem CDC CAM.

Ukázku přidaného hardwaru a robota samotného lze vidět na obrázku 11. Díky malým rozměrům přidaného routeru bylo možné toto zařízení společně s kabeláží nahradit místo prachového filtru v horní části robota. Z důvodu odstranění prachového filtru z robotického vysavače je proto nevhodné spouštět funkce vysávání.

Obrázek 11: Neato XV-15 s přidaným Wi-Fi routerem

6 Android ovládací a vizualizační aplikace

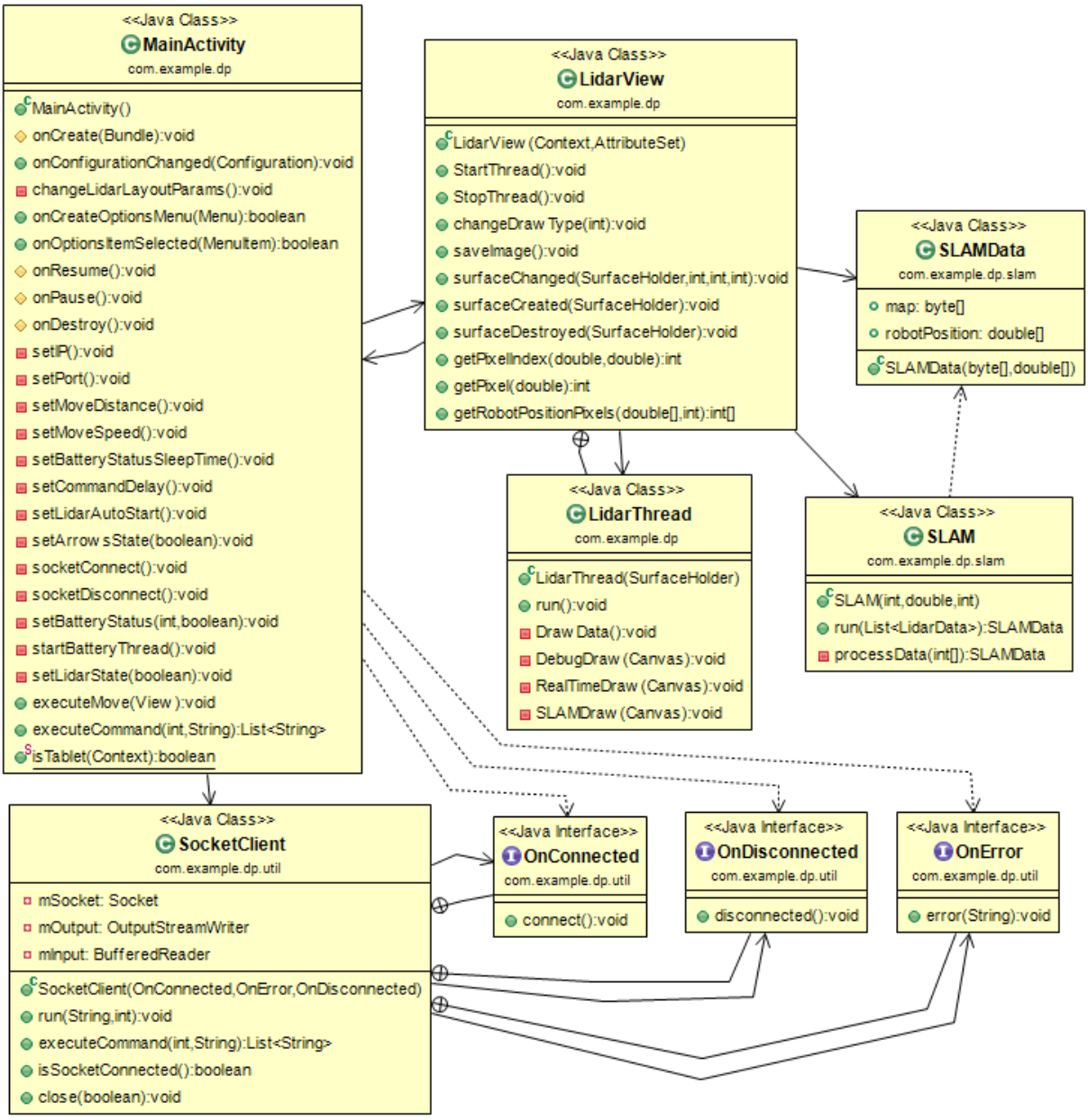
Aplikace byla původně tvořena pro Android od verze 2.2 a novější. Používala tedy API 8, avšak s postupem času byla tato starší verze Androidu stále méně oblíbenou a používanou. Novější verze API, především od verze 14, nabízely mnoho užitečných funkcí a nových přístupů, díky kterým se začaly rychle rozšiřovat. Aplikace tedy byla předělána pro využívání modernějšího přístupu a její aktuální minimální požadavek pro běh je API 15. Kompilace je však prováděna nyní nejnovějším dostupným API 21, tedy neměl by být problém spustit aplikaci na jakémkoliv moderním Android zařízení. Dále je použito nejnovější dostupné NDK r10d pro vykonávání nativního C / C++ kódu.

6.1 Implementace

Hlavní třídou obstarávající chod aplikace je *MainActivity*. Tato třída propojuje jiné použité třídy a komponenty, reaguje na vybrané akce z menu aplikace a je základem pro vlákna starající se např. o komunikaci s robotem, zjišťování stavu baterie a zpracování a vykreslování LIDAR dat. Ve třídě *MainActivity* nalezneme metodu *socketConnect* pro vytvoření socket komunikace a metodu *socketDisconnect* pro ukončení socket komunikace. Důležitou metodou je *executeCommand*, která slouží k zaslání příkazu robotovi a dále metoda *startBatteryThread* pro spuštění nového vlákna, které se bude starat o zjišťování stavu baterie robota.

Třída *MainActivity* je úzce spjatá se třídou *LidarView* a třídou *SocketClient*. Třída *LidarView* je v hlavní třídě načtena pomocí *View* komponenty a stará se o úkony spojené s vizualizací dat. Třída *SocketClient* je určená k realizaci socket spojení a komunikaci s robotem. Tato třída má přímý vliv na chování aplikace a to pomocí rozhraní *OnConnected*, *OnError* a *OnDisconnected*, které informují třídu *MainActivity* o stavu spojení s robotem.

Základní přehled tříd a metod lze vidět na obrázku 12.



Obrázek 12: Třídní diagram základních tříd aplikace

6.1.1 Komunikace s robotem

Vytvořená aplikace komunikuje s robotem skrze socket připojení. Implementace skrze socket spojení se nachází ve třídě *SocketClient*. V konstruktoru této třídy se inicializují rozhraní pro informování hlavní třídy *MainActivity* o stavu spojení s robotem. Pro vytvoření socketu slouží metoda *run*, které se v parametrech předá IP adresa robota a port, přes který bude spojení realizováno. Nastavení IP adresy a portu lze najít v nastavení aplikace.

Připojení aplikace k robotovi má na svědomí metoda *socketConnect* nacházející se ve třídě *MainActivity*. V metodě se vytvoří nové vlákno, které se postará o samotné připojení k robotovi. Časový

limit pro připojení je nastaven na patnáct sekund. Pokud se do tohoto časového limitu nepodaří úspěšně navázat spojení, bude aplikace o této skutečnosti uživatele informovat pomocí chybové hlášky.

Po úspěšném připojení socketu se vytvoří objekt ze třídy *OutputStreamWriter* pro odesílání dat a dále objekt ze třídy *BufferedReader* pro přijímání dat. Posléze se zašle robotovi příkaz „*TestMode on*“ pro aktivaci testovacího režimu, který je nutný pro plné ovládání robota a pro práci s laserovým zařízením.

Hlavní vlákno aplikace je informováno o úspěšně navázaném spojení za účelem provedení příslušných změn v uživatelském rozhraní, jako je zpřístupnění ovládacích tlačítek pro pohyb s robotem, možnost aktivování laserového zařízení, zobrazení posledního provedeného příkazu apod.

6.1.2 Zasílání příkazů a čtení odpovědí

Pro provedení požadovaných akcí je zapotřebí robotovi zasílat dostupné příkazy, kterým robot rozumí. Daný příkaz může a nemusí vracet odpověď, nicméně na každý zaslaný příkaz se vrací odpověď minimálně ve stejném znění, jako byl zadaný příkaz. Odpověď je v CSV formátu a je zakončená znakem control-Z (^Z). [35]

K zaslání příkazu a následnému přečtení odpovědi slouží metoda *executeCommand*. Jedná se o synchronizovanou metodu a to z toho důvodu, aby se předešlo zasílání více příkazů najednou ještě před tím, než je robot stihne zpracovat. V metodě se tedy zašle příkaz a ihned se čeká na odpověď, kterou má robot zaslat zpět. Pro úspěšné volání metody je nutné zadat dva parametry. Prvním parametrem je počet řádků odpovědi, které má robot zaslat zpět a druhým parametrem je příkaz samotný. Ze vstupu se vždy přečte řádek, pokud první načtený řádek odpovídá zaslanému příkazu, začnou se odpočítávat řádky, které k příkazu přísluší a načtené řádky se postupně ukládají do listu, který je nakonec vrácen jako výsledek metody. V případě, kdy první načtený řádek neodpovídá zaslanému příkazu, metoda pokračuje v načítání dat, dokud se tomu tak nestane anebo nevyprší pětisekundový časový limit pro čtení dat. Toto ověření a časový limit zde musí existovat, jelikož se mnohokrát stalo, že robot přestal odpovídat a nebýt této kontroly, aplikace by zamrzla nebo by mohlo dojít k pádu aplikace z důvodu načtení nekonzistentních dat apod.

Je tedy žádoucí vždy načíst veškerá potřebná data a v případech, kdy se tomu tak nestane, vhodně reagovat na nastalou situaci, informovat o tom uživatele a pokusit se z této situace zotavit. Z důvodu „zamrznutí“ robota, kdy robot nestačil odeslat celou odpověď a posléze přestal úplně reagovat na jakýkoliv další podnět, byla do aplikace přidána kontrola. Pokud robot dvakrát za sebou neodpoví na zaslaný příkaz, aplikace automaticky spojení ukončí.

Dále po testování a hledání možné příčiny, proč robot přestane po nějaké době komunikovat, se zjistilo, že pokud se robotovi zasílá příliš hodně příkazů za sebou, robot zamrzne a nelze s ním poté již nic dělat. Proto se v nastavení aplikace nachází možnost nastavit prodlevu mezi zasílanými příkazy.

6.1.3 Ukončení komunikace

Ještě před ukončením spojení je vždy zapotřebí řádně přepnout robota do jeho běžného stavu. Robot se v testovacím stavu sám nevypne a stále čeká na příchozí příkazy, proto je nutné mu ještě před uzavřením socketu zaslat příkaz „*TestMode off*“ pro vypnutí testovacího módu. Pokud byla zapnuta LIDAR funkce, je taktéž zapotřebí tuto funkci robota vypnout, aby se laserové zařízení přestalo točit. Poté již může být komunikace bezpečně ukončena.

6.1.4 Stav baterie robota

Kontrola stavu baterie připojeného robota se kontroluje opakovaným zasíláním příkazu v samostatném vlákně aplikace. Toto vlákno je vytvořeno v hlavní třídě *MainActivity* a interval kontroly stavu baterie se načítá z nastavení aplikace. K vizuální reprezentaci stavu baterie robota je použito šestnáct standardních ikonek dostupných na Android platformě. Konkrétně osm pro reprezentaci hodnoty baterie při stavu nabíjení baterie a osm naopak při běžném stavu bez nabíjení baterie.

6.1.5 Pohyb robota

Pro uvedení robota do pohybu se robotovi zasílají příslušné příkazy, které aktivují hnací motorky kol. Pohyb robota se provádí krokově, tedy robotovi se v příkazu zašlou nezbytné údaje o vzdálenosti a rychlosti pohybu. Funkcionalita pohybu je implementována v hlavní třídě *MainActivity*.

Pohyb s robotem v reálném čase byl ze začátku taktéž implementován, nicméně toto řešení bylo odstraněno po zjištění, že robot po nějaké době sám od sebe přestane reagovat a přijímat jakékoliv další příkazy, což vedlo k nemožnosti robota jakkoliv zastavit. A to proto, že ovládání v reálném čase bylo možné provést pouze pomocí zaslaného příkazu pro zahájení otáčení kol, které trvá do chvíle, než se robotovi zašle další příkaz pro zastavení pohybu kol. Pokud tedy robot nezpracuje námi zaslaný příkaz pro zastavení kol, tak pokračuje dále v pohybu, což bylo nežádoucí.

6.1.6 Zpracování a vizualizace LIDAR dat

Třída *LidarView* společně s vnitřní třídou *LidarThread* jsou určeny k řešení úkonů spojených s vizualizací dat. Třída *LidarView* je zde pro možnost komunikace s hlavní *MainActivity* třídou, ale zároveň řídí vlákno obstarávající metody pro získání, zpracování a vykreslení LIDAR dat, které jsou implementovány právě ve třídě *LidarThread*. Ve třídě *LidarThread* tak nalezneme metodu *DebugDraw* pro vykreslení dat v „Debug“ režimu, dále metodu *RealTimeDraw* určenou pro vykreslování dat v „Real-time“ režimu a metodu *SLAMDraw*, která se postará o vykreslení LIDAR dat ve „SLAM“ režimu.

Získávání LIDAR dat z robota se provádí zasíláním příkazu „*GetLDSScan*“ v metodě *run* třídy *LidarThread*. Dokud je LIDAR funkcionalita v aplikaci zapnuta, ve smyčce se neustále opakuje načítání dat a jejich následné zpracování a vykreslení. Jakmile je tato funkcionalita vypnuta, dojde k zaslání příkazu pro vypnutí laserového zařízení robota.

Vykreslování dat se provádí do klasické Android *View* komponenty, dědící funkcionalitu ze třídy *SurfaceView*. Samotné kreslení se provádí pomocí *Canvas* rozhraní. Ještě před samotnou

vizualizací dojde k parsování dat získaných od robota. Data jsou uložena v listu, kde každý záznam reprezentuje právě jeden úhel. Zpracováváno je vždy celých 360 stupňů. Ke zpracování každého záznamu byla vytvořena třída *LidarData*, které se parametrem zašle získaný řádek zaslaný robotem. Ten se zpracuje a dále je již poté možné objektově přistupovat k jednotlivým informacím, kterými jsou úhel, vzdálenost, intenzita a chybový kód.

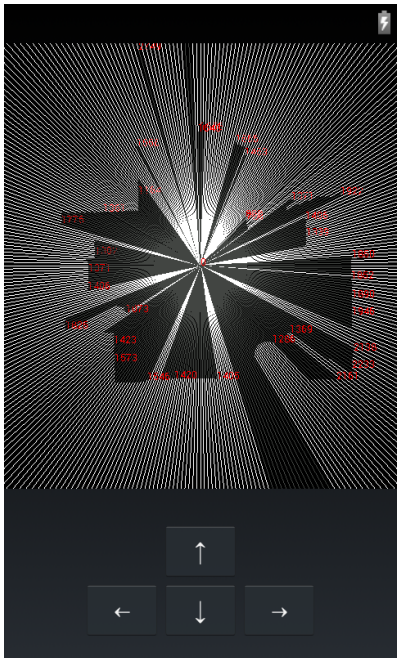
K provádění lokalizace a mapování je vytvořena ve třídě *LidarView* instance třídy *SLAM* a taktéž je zde inicializován objekt *slamTrajectory* pro ukládání navštívených pozic robotem. Pro vytvoření objektu *slam* ze třídy *SLAM* se v parametrech konstruktoru předávají konstanty *SLAM_MAP_SIZE*, reprezentující velikost mapy v pixelech, a *SLAM_MAP_SIZE_METERS*, reprezentující měřítko mapy v metrech. Hodnoty těchto konstant lze nalézt a změnit ve třídě *Consts*.

V případě zpracovávání SLAM operací se využívá třída *SLAMData* pro uchovávání vygenerované mapy a aktuálního odhadu pozice robota. S těmito informacemi se dále pracuje v případě posouvání mapy uživatelem, či ukládání mapy do souboru.

6.1.6.1 Debug režim

V tomto režimu jsou vykreslována veškerá aktuálně získaná data, tedy i ta, která jsou zatížené chybou, nejsou přesná apod. Nedochází k žádnému rozlišování intenzity a spolehlivosti údaje, zkrátka s údaji se pracuje přesně v takové formě, v jaké byly získány. Vykreslování prostoru se provádí pomocí úseček a to tak, že se vypočte podle úhlu a vzdálenosti detekované překážky její pozice pro vykreslení a od tohoto bodu je vykreslena úsečka až do konce vykreslovaného prostoru (hranice *Canvasu*). Tedy volný prostor, kde nebyly detekovány žádné překážky, je reprezentován prázdným prostorem na displeji. Okolí, které již robot nevidí, je prezentováno vykreslenými úsečkami. U překážek se v tomto „Debug“ režimu vypisuje naměřená vzdálenost a to pouze tehdy, je-li pro tento údaj na dané pozici volné místo.

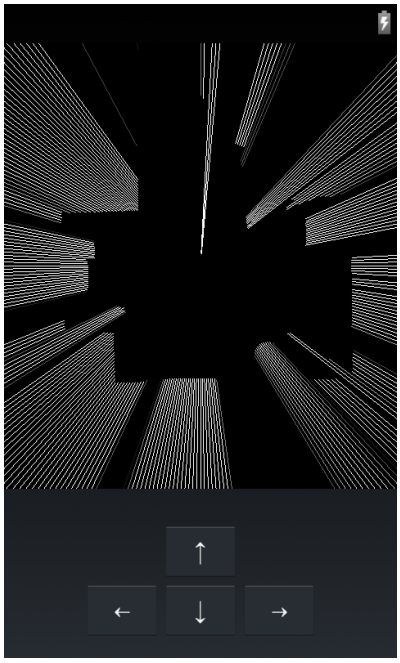
Ukázku vizualizace dat v „Debug“ režimu lze vidět na obrázku 13.



Obrázek 13: Ukázka „Debug“ režimu vykreslování dat

6.1.6.2 Real-time režim

Režim „Real-time“ vykresluje pouze data, která jsou aktuálně dostupná, tedy nepracuje s žádnými předchozími daty a výsledkem je přehled aktuálního stavu okolí jako v případě „Debug“ módu. Ukázku výsledku vykreslování v tomto režimu můžeme vidět na obrázku 14. Nalezené překážky jsou opět reprezentovány čárami tak, že z místa detekované překážky vychází čára směrem ven do prostoru reprezentující místo, kam robot nevidí. Jednotlivé čáry jsou však v tomto režimu odlišené šedým odstínem podle naměřené intenzity, tedy jinak řečeno podle spolehlivosti naměřeného údaje. Tmavší odstín šedé značí spolehlivější údaj. Nepřesné údaje s nízkou intenzitou a údaje zatížené chybou při skenování nejsou v tomto režimu vykreslovány.



Obrázek 14: Ukázka „Real-time“ režimu vykreslování dat

6.1.6.3 SLAM režim

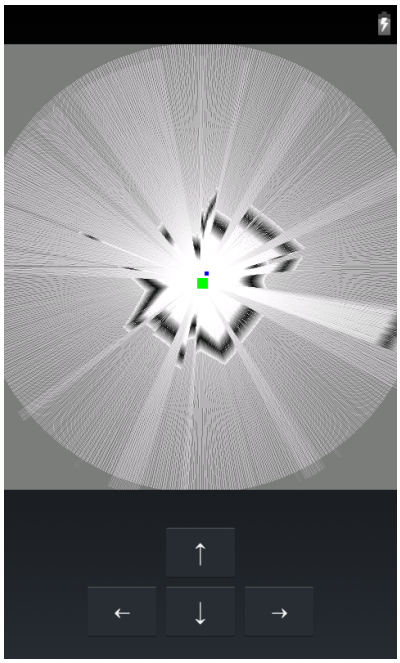
Jak již název napovídá, tento režim slouží pro řešení simultánní lokalizace a mapování a to za pomoci zvoleného BreezySLAM řešení. Data jsou jako v obou přechozích režimech nejdříve zpracována třídou *LidarData* a posléze se již o zpracování lokalizace a mapování stará samotný algoritmus. BreezySLAM řešení po zpracování dat vrací jak vytvořenou mapu, tak i trajektorii (pozici robota) v mapě. Pro vykreslování kompletní trajektorie, kterou robot urazil, je tedy zapotřebí si tento údaj ukládat např. do listu, jak je učiněno zde. Po zpracování veškerých potřebných údajů dojde k vytvoření pole jednotlivých pixelů, které vytvoří celkovou bitmapu použitou pro následné vykreslení. BreezySLAM používá pro reprezentaci překážek pouze odstíny šedé barvy a veškerá data by za normálních podmínek byla uložena do PGM souboru. To však pro nás není žádoucí, jelikož takový výstup neobsahuje žádné informace o trajektorii robota a ani není zapotřebí data ukládat do souboru, jelikož je v našem případě chceme zobrazit na displeji mobilního zařízení. Z tohoto důvodu jsou nejdříve zpracovány všechny data reprezentující jednotlivé pixely naší vytvořené mapy, poté se změní barva příslušných pixelů, reprezentujících trajektorii robota, na modrou. Dále se změní pixely náležící aktuální pozici robota na zelené a až nakonec dojde k vytvoření kompletní bitmapy, která je poté vykreslena.

Protože většina mobilních zařízená má nižší rozlišení, než jaké má vykreslovaná bitmapa, tak je možné s vytvořenou mapou na displeji pohybovat. Celkové rozlišení mapy a její měřítko je možné měnit v kódu ve třídě *Consts*, konkrétně tedy konstantu *SLAM_MAP_SIZE* reprezentující výšku a šířku mapy v pixelech a dále konstantu *SLAM_MAP_SIZE_METERS* reprezentující velikost oblasti, která má být skenována, v metrech.

Další důležité nastavení se nachází ve třídě *SLAM*. V konstriktoru třídy se vytváří objekt *Laser*, kterému je nutné nastavit počet snímaných stupňů, což je v tomto případě 360 stupňů. Dále frekvence

snímání, která je nastavená na 10 Hz a v neposlední řadě maximální snímaná vzdálenost nastavená na 5 m.

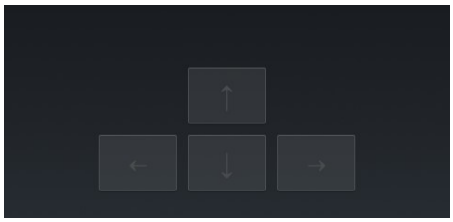
Ukázku „SLAM“ vizualizačního režimu lze vidět na obrázku 15, kde vytvořená mapa reprezentuje mapování kanceláře.



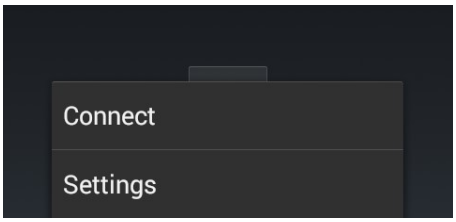
Obrázek 15: Ukázka „SLAM“ režimu vykreslování dat

6.2 Uživatelské rozhraní

Po spuštění aplikace vidí uživatel pouze zašedlá tlačítka pro ovládání pohybu robota, která lze vidět na obrázku 16. V případě zobrazení nabídky menu jsou dostupná dvě tlačítka a to „Connect“ pro navázání spojení s robotem a „Settings“ pro zobrazení obrazovky s nastavením aplikace. Ukázka základní nabídky menu lze vidět na obrázku 17.



Obrázek 16: Neaktivní tlačítka na úvodní obrazovce

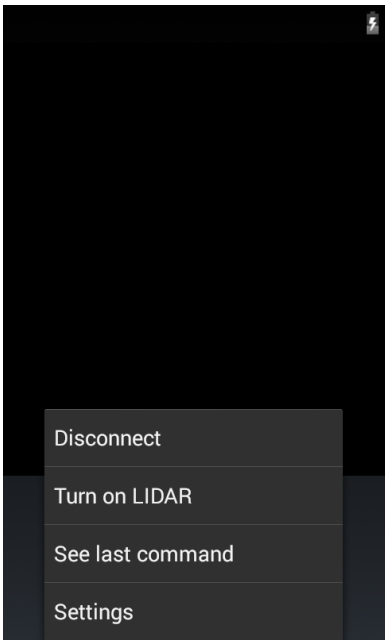


Obrázek 17: Základní možnosti v menu aplikace

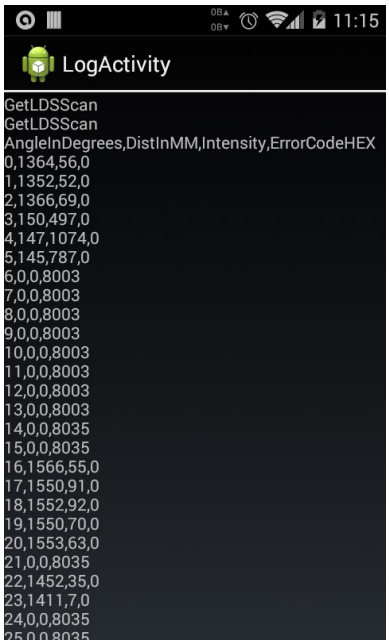
Po úspěšném připojení aplikace s robotem v menu přibydou další možnosti aplikace, které lze vidět na obrázku 18. První přidanou možností je zapnutí / vypnutí LIDAR technologie (v menu položka „Turn on LIDAR“ / „Turn off LIDAR“), kdy po zapnutí této funkce dojde k vytvoření nového vlákna aplikace, které se stará jak o zpracování a vizualizaci dat, tak i jejich získávání.

Druhou přidanou položkou je „See last command“, která slouží k zobrazení posledního provedeného příkazu a odpovědi na něj. Z obrázku 19 lze vidět příklad záznamu posledního provedeného příkazu. Další položkou v menu je „DrawType“, která je dostupná pouze v případě, že je zapnuta LIDAR funkcionalita. Tato položka umožňuje změnit styl zpracování a zobrazování získaných LIDAR dat, která se následně vizualizují. Mezi dostupné možnosti režimů vizualizace dat patří „SLAM“, „Real-time“ a „Debug“.

Poslední možností, která je v menu přítomna pouze v případě zapnuté LIDAR technologie, je „Save to image“. Kliknutím na tuto položku dojde k uložení naposledy vytvořené mapy pomocí SLAM technologie. Obrázek je uložen do předem nastavené složky DP a název ukládaného souboru obsahuje čas uložení.

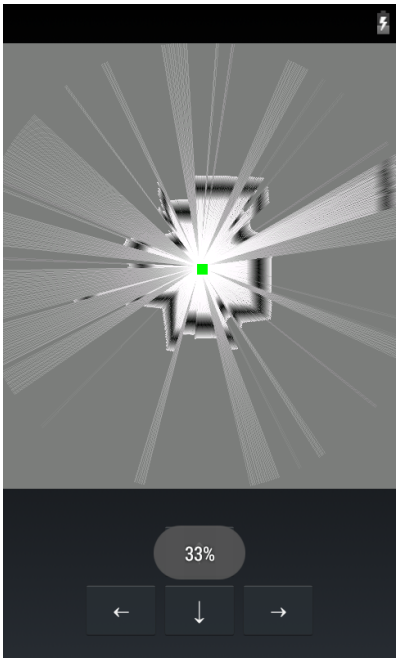


Obrázek 18: Ukázka menu a ikonky baterie po úspěšném připojení k robotovi

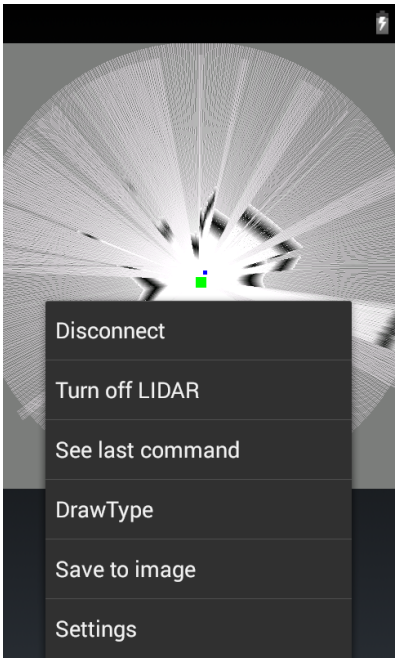


Obrázek 19: Ukázka posledního zaslaného příkazu a odpovědi na něj

V případě, že uživatel chce vědět přesnou hodnotu stavu baterie, stačí kliknout na ikonku baterie umístěné v pravém horním rohu a v oznamovací části se ukáže číselná hodnota, jak lze vidět na obrázku 20. Veškeré dostupné možnosti v menu aplikace s úspěšně navázaným spojením s robotem a zapnutou LIDAR technologií lze vidět na obrázku 21.



Obrázek 20: Ukázka zobrazení stavu baterie robota po kliknutí na ikonku baterie

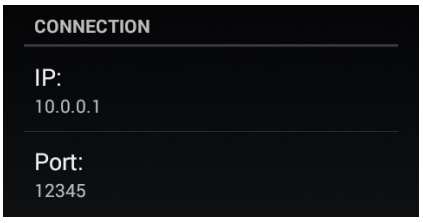


Obrázek 21: Dostupné možnosti v menu aplikace se zapnutou LIDAR technologií

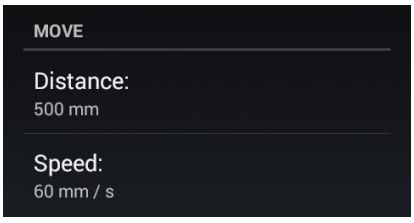
6.2.1 Nastavení

Před prvním připojením robota je zapotřebí aplikaci nastavit. Pokud tak uživatel neučinil již sám v nastavení aplikace, aplikace při zjištění nenastavených povinných údajů k úspěšnému připojení sama pobídne uživatele k jejich vyplnění a to otevřením obrazovky s nastavení aplikace. Mezi povinné údaje pro připojení k robotovi patří IP adresa robota, na kterou se aplikace má připojit a dále číslo portu, přes který bude komunikace probíhat. Ukázku jak tato sekce vypadá v nastavení aplikace, můžeme vidět na obrázku 22.

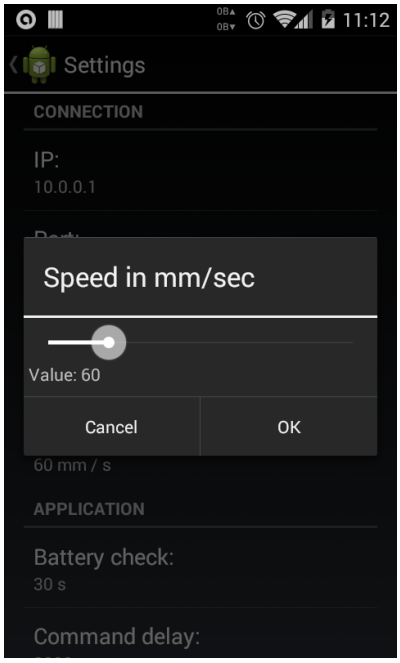
Dalšími položkami v nastavení jsou „Distance“ a „Speed“, které slouží k nastavení pohybu robota. Tuto sekci v nastavení můžeme vidět na obrázku 23. První jmenovaná položka slouží k nastavení vzdálenosti (v milimetrech), kterou robot má urazit po zmáčknutí příslušného tlačítka pro pohyb robota. Druhá možnost určuje rychlost (v milimetrech za sekundu), kterou se bude robot pohybovat. Pro nastavení rychlosti pohybu robota byla vytvořená vlastní komponenta posuvníku. Díky tomu je nastavení rychlosti jednoduší a je také zabráněno nastavení příliš vysoké hodnoty. Ukázku tohoto řešení lze vidět na obrázku 24.



Obrázek 22: Sekce připojení v nastavení aplikace

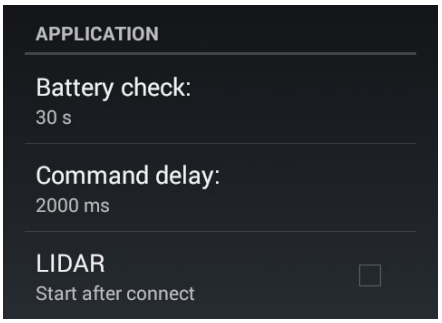


Obrázek 23: Sekce pohybu robota v nastavení aplikace



Obrázek 24: Ukázka posuvníku pro nastavení rychlosti pohybu robota

V poslední části nastavení aplikace se nachází položka „Battery check“ pro nastavení, jak často se má kontrolovat stav baterie robota (hodnota je uvedena v sekundách). Dále položka „Command delay“ k nastavení časové prodlevy (v milisekundách) mezi zasílanými příkazy a jako poslední se zde nachází zaškrtávací políčko „LIDAR“ pro nastavení, zda se má LIDAR funkcionalita zapínat ihned po navázání spojení s robotem, či nikoliv. Poslední sekci v nastavení aplikace můžeme vidět na obrázku 25.



Obrázek 25: Sekce nastavení aplikace

6.2.2 Ovládání pohybu robota

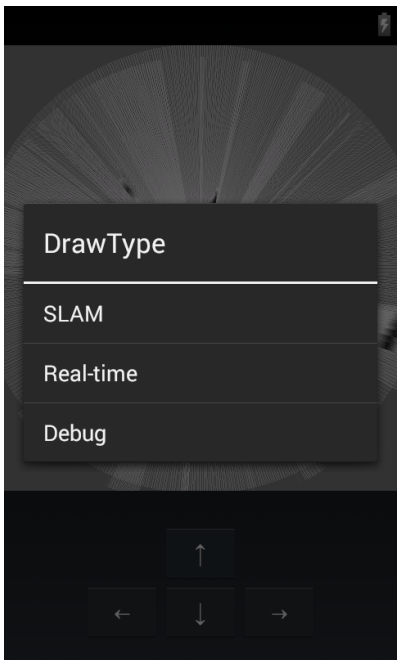
Po úspěšném navázání spojení s robotem se neaktivní zašedlá tlačítka, sloužící k pohybu s robotem, změní na aktivní tlačítka, která reagují na kliknutí. Po kliknutí na tlačítko se zašle příslušný příkaz robotovi pro aktivaci motorů kol a tím se docílí pohybu robota. Pohyb se provádí krokově, tedy robotovi se v příkazu zašle jakou vzdálenost a při jaké rychlosti má pohyb provést. Není tedy možné ho

ovládat v reálném čase, pokud klikneme na tlačítko vícekrát za sebou, robot vždy provede příkazy v plném rozsahu, tak jak je přijal.

6.2.3 Vizualizace okolí

Pro vizualizaci dat získaných z LIDAR technologie je vyhrazena horní část obrazovky. Tuto technologii lze zapnout pomocí možnosti „Turn on LIDAR“, nacházející se v menu aplikace. Případně může být tato technologie zapnuta automaticky a to v případě, pokud je v nastavení aplikace zaškrtnuta možnost „LIDAR - Start after connect“.

V případě, že je LIDAR technologie zapnuta, menu aplikace obsahuje položku „DrawType“, která po kliknutí otevře výběr dostupných režimů vykreslování. Ukázku výběru režimů lze vidět na obrázku 26. K dispozici jsou tři režimy vizualizace a to „SLAM“, „Real-time“ a „Debug“. V případě „SLAM“ režimu je možné vykreslovanou mapou posouvat.



Obrázek 26: Dostupné režimy pro vykreslování získaných LIDAR dat

7 Testování a zhodnocení řešení

Testování aplikace probíhalo jak na reálném robotovi, tak i pomocí komunikace s vlastní vytvořenou testovací Java aplikací, která sloužila pro simulaci komunikace s robotem. Tato jednoduchá testovací aplikace vytvoří socket rozhraní na předem definovaném portu a posléze již jen čeká na připojení klienta. Po přečtení zaslaného příkazu zašle příslušnou odpověď tak, jak by provedl reálný robot. Pro korektní testování LIDAR technologie dokáže testovací aplikace načíst data z textového souboru a to proto, aby se neustále nezasílala pouze jedna předdefinovaná odpověď, ale aby se reflektovaly skutečné naměřené údaje. Načítaná data byla předem získána z reálného robota, tedy nejedná se o žádné náhodně generované hodnoty.

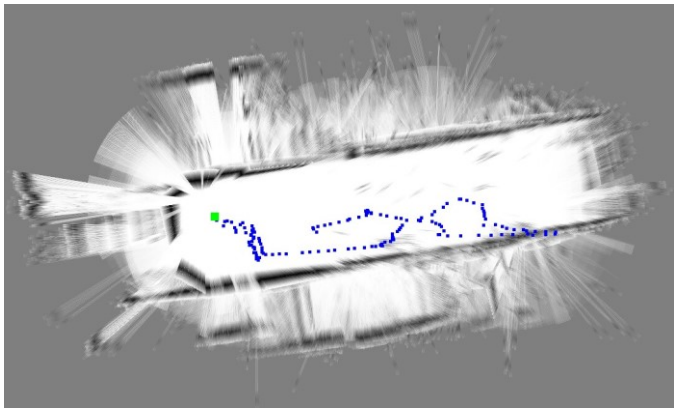
Vytvořená aplikace byla dále testována jak na reálném mobilním zařízení, tak i na oficiálně dostupných Android emulátorech. Použité emulátory byly založené na ARM systémovém obrazu a také akcelerovaném x86 systémovém obrazu.

Důkladné testování s robotickým vysavačem Neato XV-15 odhalilo zásadní problémy, se kterými se bylo zapotřebí vypořádat. Prvním hlavním problémem bylo samovolné vypínání robota. Po náhodném čase se robot z ničeho nic vypínal, případně přestal reagovat na jakékoliv zasílané příkazy a potřeboval ruční zásah, jako bylo odpojení USB portu nebo případně plné restartování robota. Z tohoto důvodu byla odstraněna možnost kontinuálního pohybu robota, aby nedošlo k případu, kdy by robota nebylo možné zastavit jinak než ručním zásahem.

Druhým závažným problémem je možné zahlcení robota častým zasíláním příkazů. Především pro zajištění korektního fungování SLAM technologie je časté čtení dat z laserového zařízení velice zásadní a je žádoucí data získávat co nejčastěji je možné. Avšak v případě tohoto robotického vysavače nastal s častým zasíláním příkazů problém. Z tohoto důvodu bylo do aplikace přidáno nastavení prodlevy mezi zasílanými příkazy. Tato prodleva může být v nastavení aplikace změněna, avšak defaultní hodnota byla nastavena na dvě sekundy. Hodnota dvou sekund se ukázala být stabilním řešením i pro dlouhodobé používání robota, avšak to bohužel s sebou přineslo zhoršení fungování simultánní lokalizace a mapování.

Samotné hodnocení použitého SLAM řešení je spíše subjektivní a to z důvodu absence možnosti srovnání vygenerovaných výsledků s jiným řešením. Nicméně porovnání použitého SLAM algoritmu s jinými algoritmy lze nalézt ve zdrojích [32, 36].

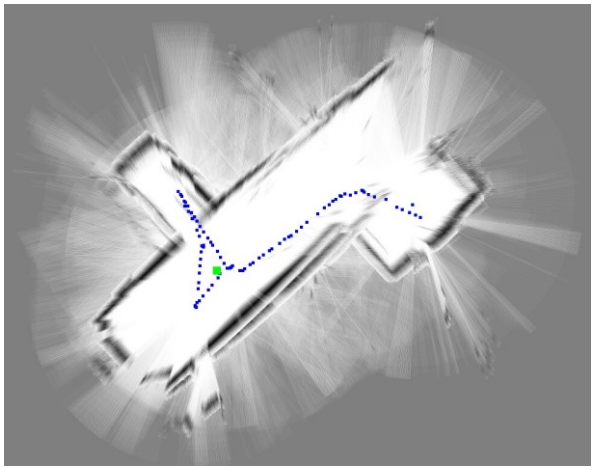
Na obrázku 27 lze pozorovat nedokonalost SLAM řešení. Algoritmus má problém s uzavřením smyček a proto překresluje aktuálně naskenovaný prostor v jiném naklonění, než ve kterém skutečně je. Tato skutečnost je také hlavně příčinou dlouhé prodlevy mezi jednotlivým skenováním okolí. Pokud se provede skenování a mezi dalším skenováním, které se provede v ideálním případě za další dvě sekundy (podle nastavené prodlevy příkazů), se robot posune o delší vzdálenost a v horším případě se i natočí jiným směrem, algoritmus má těžkou úlohu odhadnout aktuální pozici robota a jeho natočení. Výsledek takové skutečnosti, kdy bylo robotem neustále otáčeno dokola, je právě na obrázku 27 znázorněn.



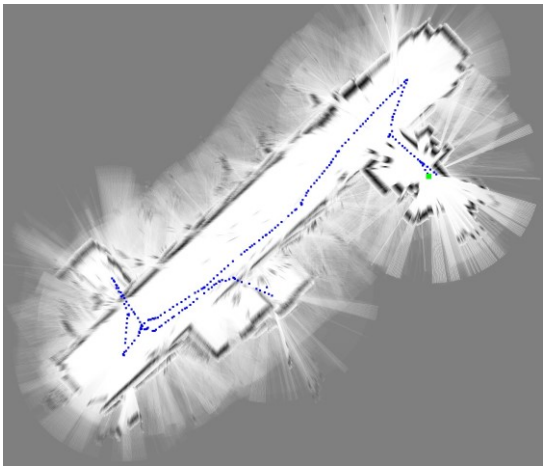
Obrázek 27: Ukázka problému mapování naskenovaného prostoru

Na obrázku 28 můžeme vidět vygenerovanou mapu ze začátku průjezdu robota chodbou. Mapa se generovala v pořádku do chvíle, než robot přestal s aplikací komunikovat. Při opětovném navázání spojení a spuštění LIDAR technologie došlo k mírnému posunu odhadnuté pozice robota v mapě, což zapříčinilo posun levé a pravé dolní stěny. Po otočení robota a jízdy na opačný konec chodby došlo několikrát k zaseknutí robota. Opětovně spuštěné SLAM řešení se dokázalo poměrně přesně zorientovat v okolí a odhadnout pozici robota v naposledy vygenerované mapě, avšak jak lze vidět na obrázku 29, vždy došlo k posunutí vykreslovaného prostoru. Nejvíce jde tato skutečnost vidět na počátečním místě, kde robot začínal své skenování. Skenováním prostoru bylo ukončeno v jedné z kanceláří, která byla algoritmem poměrně dobře zmapována.

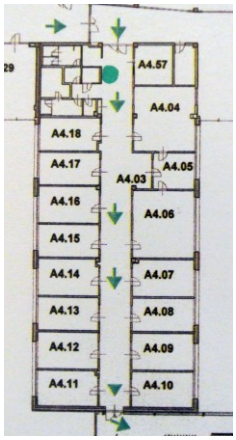
Obrázky 30 a 31 zachycují reálný stav chodby, kterou robot projížděl a snímal (výsledná mapa na obrázku 29). Šířka chodby, společně s výklenkem dveří, na vygenerované mapě činí přibližně 325 cm (měřeno v horní části mapy), zatímco reálná naměřená šířka činila 328 cm. Samotná šířka chodby, bez výklenků, na vygenerované mapě činí přibližně 296 cm a to vesměs po celé její délce, což ukazuje na poměrně přesné zmapování prostoru s přijatelnou odchylkou.



Obrázek 28: Ukázka vygenerované mapy



Obrázek 29: Ukázka mapy vytvořené průjezdem robota chodbou do kanceláře



Obrázek 30: Ukázka plánku chodby

Obrázek 31: Chodba

Delší prodlevy mezi snímáním okolí v uzavřeném prostoru jako jsou chodby a kanceláře nepředstavuje pro tento SLAM algoritmus příliš velké problémy. Avšak je třeba neopomenout skutečnost, kdy ve výsledné mapě může chybět např. celá boční chodba a to v případě rychlého pohybu robota a dlouhých prodlev snímání okolí. Robot může těsně před začátkem boční chodby provést měření, pokračovat v pohybu a další měření nastane až za touto boční chodbou. I kdyby stihnul robot provést měření uprostřed křížení chodeb (či výklenku apod.), výsledná mapa nemusí tuto skutečnost reflektovat a to proto, že algoritmus potřebuje více měření pro reflektování skutečného stavu okolí. Díky tomuto postupnému zakreslování objektů lze snadněji odstranit nežádoucí objekty, kterými mohou být pohybující se objekty, jako jsou např. lidé. Při testování robota bylo na mapě zřetelné, když robot detekoval osobu na chodbě a tuto překážku do mapy zakreslil a s dalšími provedenými iteracemi postupně překážka z mapy mizela.

Největším zjištěným problémem BreezySLAM řešení byl případ, kdy došlo k zaseknutí robota s přerušením komunikace a bylo tedy zapotřebí znovu zjistit polohu robota z dříve vygenerované mapy. Algoritmus sice dokázal, poměrně rychle, odhadnou aktuální pozici robota v mapě, avšak většinou došlo k patrnému posunu vykreslovaného prostoru, což ve vizualizaci zapříčiňovalo zdvojení zdí, překreslování zdí do volného prostoru apod.

Rychlost zpracování SLAM algoritmů je rychlé, v případě nulové prodlevy mezi zasílanými příkazy je běh téměř plynulý. S přihlédnutím na skutečnost, že testování bylo prováděno na poměrně starém mobilním telefonu, je rychlost více než dostačující. Proto i s ohledem na poměrně nízké paměťové nároky bylo zvoleno právě BreezySLAM řešení jako nejvhodnější pro použití na mobilním zařízení. Ještě před použitím BreezySLAM řešení bylo v aplikaci otestováno vlastní řešení GraphSLAM algoritmu. Vlastní implementace GraphSLAM algoritmu byla provedena v jazyce Java. Po prvních provedených testech bylo však rozhodnuto nepokračovat v implementaci tohoto řešení a to z důvodu vyšších paměťových nároků a celkem pomalého zpracování jednoho cyklu. V tomto ohledu tedy vlastní implementace řešení v jazyce Java nedokáže konkurovat mnohem rychlejšímu algoritmu v jazyce C, zpracovávaném v nativním režimu systému Android.

Vylepšením realizovaného SLAM řešení by bylo přidání algoritmu pro uzavírání smyček, což by pomohlo k přesnějšímu mapování prostoru a odhadu pozice robota. Taktéž by k lepšímu fungování

algoritmů přispěli informace z GPS a kompasu, které by sloužili k upřesnění trajektorie robota. Použití mnohem sofistikovanějšího řešení by v dnešní době již taktéž neměl být problém. Oproti starému mobilnímu zařízení, které bylo zde použito pro testování, mají dnešní mobilní zařízení vyšší výkon a dostatek operační paměti, aby byly schopné provozovat mnohem paměťově a výkonnostně náročnější řešení.

8 Závěr

Cílem diplomové práce bylo vytvořit Android aplikaci, která bude schopná ovládat robotickou platformu a vizualizovat data získaná z laserového zařízení. Dále mělo být v aplikaci implementováno řešení pro simultánní lokalizaci a mapování.

Začátek diplomové práce obsahuje základní přehled robotických platforem a komunikačních rozhraní. Přehled platforem obsahuje jak oblíbené a známé roboty, tak i jedny z nejnovějších robotů inspirovaných přírodou. Jedním z nejpopulárnějších vzdušných robotů je AR.Drone 2.0. Oblíbenost si zasloužil díky jednoduchému, intuitivnímu ovládání a svému technickému zpracování. Jako nové robotické platformy jsou zde uvedeni roboti BionicANTs a eMotionButterflies. V obou případech se jedná o přírodou inspirované roboty, konkrétně BionicANTs se podobají mravencům a eMotionButterflies motýlům. Stejně jako u vzhledu konstrukce je zde snaha o implementaci reálných vzorců chování. Dále je v práci zmíněn pokročilý humanoidní robot Atlas, který může sloužit jako ukázka nejnovějších a nejmodernějších technologií v robotické oblasti. Jako příklad hotového robotického řešení je zde uveden robot Roli, který je schopen autonomního chování, vizuálního učení, plnění hlasových příkazů, detekování a sledování objektů atd.

V práci je dále popsána LIDAR technologie společně s řešeními pro simultánní lokalizaci a mapování. Popis se soustředí převážně na CoreSLAM a BreezySLAM řešení, která jsou ve vytvořené aplikaci využívaná k vizualizaci dat. Algoritmy v těchto SLAM řešeních jsou paměťově a výpočetně méně náročná oproti jiným, mnohem sofistikovanějším řešením. Právě z tohoto důvodu bylo BreezySLAM řešení použito ve vytvořené aplikaci pro mobilní zařízení.

Robotický vysavač Neato XV-15 byl zvolen jako zástupce robotických platforem, se kterým Android aplikace komunikuje prostřednictvím navrženého komunikačního rozhraní. Pomocí tohoto komunikačního rozhraní se robotovi zasílají ovládací příkazy a rovněž se získávají data z laserového zařízení umístěného na horní části robota.

Vytvořená Android aplikace je v práci popsána jak z implementačního hlediska, tak i z hlediska uživatelského rozhraní. Aplikace obsahuje funkcionalitu jak pro ovládání robotické platformy, konkrétně robotického vysavače Neato XV-15, tak i pro vizualizaci získaných LIDAR dat. V aplikaci je implementováno opatření pro detekování nesprávného chování robota, automatické ukončení spojení s robotem a návrat aplikace do původního stavu apod. LIDAR data lze vizualizovat pomocí implementovaného BreezySLAM řešení, které řeší lokalizaci robota v prostředí a mapování okolí.

Při implementaci aplikace se bylo zapotřebí vypořádat s různými problémy, jako např. s náhodným zamrzáváním robota. Avšak tyto problémy také dopomohly k řádnému otestování aplikace a SLAM algoritmů. Výsledkem je tedy funkční ovládací Android aplikace s lokalizačním a mapovacím algoritmem, který funguje i v předem neznámém prostředí. Aplikace pracuje plynule bez jakýchkoliv zjištěných problémů, které by způsobovaly pád aplikace. Uživatelské rozhraní je jednoduché s jednoznačným určením účelu. Všechny stanovené požadavky na aplikaci tedy byly splněny.

9 Literatura

[1] Arduino, *ArduinoBoardUno* [online]. c2015 [cit. 2015-04-29].
Dostupné z: http://arduino.cc/en/Main/ArduinoBoardUno

[2] EZ-Robot Inc., *Roli Rover* [online]. c2015 [cit. 2015-04-29].
Dostupné z: https://www.ez-robot.com/Shop/AccessoriesDetails.aspx?productNumber=32

[3] Boston Dynamics, *Dedicated to the Science and Art of How Things Move* [online]. c2013 [cit. 2015-04-29]. Dostupné z: http://www.bostondynamics.com/robot_Atlas.html

[4] Guardian News and Media Limited or its affiliated companies, *Google's massive humanoid robot can now walk and move without wires* [online]. c2015 [cit. 2015-04-29].
Dostupné z: http://www.theguardian.com/technology/2015/jan/21/googles-massive-humanoid-robot-can-now-walk-and-move-without-wires

[5] Festo, *BionicANTs. Cooperative behaviour based on a natural model*. Germany. 2015.

[6] Ministerstvo vnitra, *Portál veřejné správy* [online]. c2015 [cit. 2015-04-29].
Dostupné z: http://portal.gov.cz/app/zakony/zakon.jsp?page=0&nr=49~2F1997&rpp=15#seznam

[7] ITBIZ, *Jak si dnes stojí drony v civilních službách* [online]. c2014 [cit. 2015-04-29].
Dostupné z: http://www.itbiz.cz/clanky/droni-v-civilnich-sluzbach

[8] Parrot SA, *AR.Drone 2.0. Parrot new wi-fi quadricopter – Civil drone* [online]. c2015 [cit. 2015-04-29]. Dostupné z: http://ardrone2.parrot.com

[9] Parrot news, *AR.Drone 2.0. Even more piloting possibilities!* [online]. c2015 [cit. 2015-04-29].
Dostupné z: http://blog.parrot.com/2014/01/06/parrot-ar-drone-2-0-even-more-piloting-possibilities/

[10] DIY Drones, *The Leading Community for Personal UAVs* [online]. c2015 [cit. 2015-04-29].
Dostupné z: http://diydrones.com

[11] APM Plane, *Plane | Fixed-wing aircraft UAV* [online]. [cit. 2015-04-29].
Dostupné z: http://plane.ardupilot.com

[12] Festo, *eMotionButterflies. Ultralight flying objects with collective behaviour*. Germany. 2015.

[13] OpenROV, *OpenROV 2.7 Mini Observation Class ROV* [online]. [cit. 2015-04-29].
Dostupné z: http://www.openrov.com/products/2-7.html

[14] CBS Interactive Inc., *USB Type-C: One cable to connect them all* [online]. c2015 [cit. 2015-04-29]. Dostupné z: http://www.cnet.com/news/usb-type-c-one-cable-to-connect-them-all/

[15] Society of Robots, *I2C 101* [online]. c2015 [cit. 2015-04-29].
Dostupné z: http://www.societyofrobots.com/member_tutorials/book/export/html/35

[16] TechTarget, *What is Ethernet?* [online]. c2000-2015 [cit. 2015-04-29].
Dostupné z: http://searchnetworking.techtarget.com/definition/Ethernet

[17] QuinStreet Inc., *What is Wi-Fi (IEEE 802.11x)?* [online]. c2015 [cit. 2015-04-29].
Dostupné z: http://www.webopedia.com/TERM/W/Wi_Fi.html

[18] Bluetooth SIG, Inc., *Fast Facts | Bluetooth Technology Website* [online]. c2015 [cit. 2015-04-29].
Dostupné z: http://www.bluetooth.com/Pages/Fast-Facts.aspx

[19] USGC, *Light Detection and Ranging (LIDAR) | The Long Term Archive* [online]. c2012 [cit. 2015-04-29]. Dostupné z: https://lta.cr.usgs.gov/LIDAR

[20] National Oceanic and Atmospheric Administration, *What is LIDAR?* [online]. c2015 [cit. 2015-04-29]. Dostupné z: http://oceanservice.noaa.gov/facts/lidar.html

[21] LiDAR UK, *Find out all you need to know about LiDAR, its history, its uses and the technologies behind it* [online]. c2015 [cit. 2015-04-29]. Dostupné z: http://www.lidar-uk.com

[22] KRAUSE, K., KAMPE, T., MUSINSKY, J., *Ecological mapping. Using Integrated LiDAR and Hyperspectral Airborne Remote Sensing at NEON*, Spatial Media, 2013.

[23] Génération Robots, *Automatic Robot Vacuum Neato XV-15* [online]. [cit. 2015-04-29].
Dostupné z: http://www.generationrobots.com/en/400927-automatic-robot-vacuum-neato-xv-15.html

[24] THRUN, S., *Particle Filters in Robotics*, Morgan Kaufmann Publishers Inc., Pittsburgh, 2002.

[25] REKLEITIS, I. M., *A Particle Filter Tutorial for Mobile Robot Localization*, Montreal, Québec, 2004.

[26] CHOI, J., MEDIONI, G., *StaRSaC: Stable Random Sample Consensus for Parameter Estimation*, Los Angeles, 2009.

[27] CHEN, Z., *Bayesian Filtering: From Kalman Filters to Particle Filters, and Beyond*, Canada, 2003.

[28] WELCH, G., BISHOP, G., *And Introduction to the Kalman Filter*, Chapel Hill, North Carolina, 2006.

[29] Udacity, Inc., *Artificial Inteligence for Robotics Course* [online]. c2011-2015 [cit. 2015-04-29].
Dostupné z: https://www.udacity.com/course/cs373

[30] STEUX, B., HAMZAOUI, El O., *CoreSLAM : a SLAM Algorithm in less than 200 lines of C code*, Mines ParisTech – Center of Robotics, Paris, France, 2009.

[31] CHOWDHARY, G., SOBERS, D. M., PRAVITRA, Ch., CHRISTMANN, C., WU, A., HASTIMOTO, H., ONG, Ch., KALGHATGI, R., JOHNSON, E. N., *Self-Contained Ranging Sensor Aided Autonomous Guidance, Navigation, and Control for Indoor Flight*, Atlanta, USA, 2012.

[32] PEET, S., GROSSE, R., MORGAN, J., *Project 6 SLAM Proposal*, 2013.

[33] BreezySLAM, *BreezySLAM - A Simple, efficient, open-source package for Simultaneous Localization and Mapping in Python, Matlab, C++, and Java* [online]. c2014 [cit. 2015-04-29]. Dostupné z: http://home.wlu.edu/~levys/software/breezyslam/

[34] BAJRACHARYA, S. *BreezySLAM: A Simple, efficient, cross-platform Python package for Simultaneous Localization and Mapping*, Virginia, USA, 2014.

[35] Neato Robotics, Inc., *Programmer's Manual* [online]. c2014 [cit. 2015-04-29]. Dostupné z: http://www.neatorobotics.com/resources/programmersmanual_20140305.pdf

[36] SANTOS, J. M., PORTUGAL, D., ROCHA, R. P., *An Evaluation of 2D SLAM Techniques Available in Robot Operating System*, Portugal, 2013.

10 Seznam příloh

[1] Příloha na CD – zdrojové soubory